软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.005909]

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

人工智能赋能的数据管理新技术研究*

孙路明[1,2], 张少敏[1,2], 姬 涛[1,2], 李翠平[1,2], 陈 红[1,2]

[1](中国人民大学 信息学院,北京 100872)

[2](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)

通讯作者: 李翠平, E-mail: licuiping@ruc.edu.cn

摘 要: 大数据时代,数据规模庞大、数据管理应用场景复杂,传统数据库和数据管理技术面临很大的挑战.人工智能技术因其强大的学习、推理、规划能力,为数据库系统提供了新的发展机遇.人工智能赋能的数据库系统通过对数据分布、查询负载、性能表现等特征进行建模和学习,自动地进行查询负载预测、数据库配置参数调优、数据分区、索引维护、查询优化、查询调度等以不断提高数据库针对特定硬件、数据和负载的性能.同时,一些机器学习模型可以替代数据库系统中的部分组件,有效减少开销,如学习型索引结构等.本文分析了人工智能赋能的数据管理新技术的研究进展,总结了现有方法的问题和解决思路,并对未来研究方向进行了展望.

关键词: 数据库系统;数据管理;人工智能;机器学习;查询优化

中图法分类号: TP311

中文引用格式: 孙路明,张少敏,姬涛,李翠平,陈红.人工智能赋能的数据管理新技术研究.软件学报,2020,31(3). http://www.jos.org.cn/1000-9825/5909.htm

英文引用格式: 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) (in Chinese). http://www.jos.org.cn/1000-9825/5909.htm

Survey of Data Management Techniques Powered by Artificial Intelligence

SUN Lu-Ming[1], ZHANG Shao-Min[1], JI Tao[1], LI Cui-Ping[1], CHEN Hong[1]

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

Abstract: Traditional database systems and data management techniques are facing great challenge due to the 3V's in Big Data. The development of artificial intelligence provides a brand-new opportunity for database management systems with its power in learning, reasoning and planning. Through learning from data distribution, query workload and query execution performance, the systems powered by artificial intelligence are able to forecast future workload, tune database configurations, partition data blocks, index on proper columns, estimate selectivity, optimize query plan and control query concurrency automatically. Also, some machine learning models can replace core components of a database such as index structures. In this paper, we introduce new research on database systems with artificial intelligence and state the existing problems and potential solutions. In the end, we propose the future research directions.

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

数据库管理系统(DBMS)经过半个多世纪的研究和发展,获得了快速的技术进步和优异的性能表现.但是传统的数据库是一个静态的系统,并且更加注重系统的通用性,即不会根据历史查询性能表现自动进行数据库

* 基金项目: 国家重点研发计划(2018YFB1004401); 国家自然科学基金(61772537, 61772536, 61702522 , 61532021)

National Key Research & Develop Plan(2018YFB1004401); Foundation item: National Natural Science Foundation of China (61772537, 61772536, 61702522, 61532021)

收稿时间: 2019-07-20; 修改时间: 2019-09-10; 采用时间: 2019-11-25; jos 在线出版时间: 2019-12-05

CNKI 网络优先出版: 2019-12-05 14:55:09, http://kns.cnki.net/kcms/detail/11.2560.TP.20191205.1454.004.html

调优,也不会针对某个用户的数据和查询负载的特点进行特定的系统优化.另一方面,大数据应用场景中数据规模庞大、查询并发度高、连接操作频繁等特点对数据库系统的性能提出了更高的要求.早在上世纪九十年代初,研究者已经开始数据库的自动化管理和调优技术的探索[1,2],彼时的方法大多通过代价模型和启发式规则,进行数据库的自动调优.在工业界,以微软早期的 AutoAdmin[3]项目为代表,数据库厂商试图通过自调优技术协助用户进行数据管理,降低系统使用的难度和专业性.然而这些方法仍然受限于代价模型的准确性和人为设置启发式规则的局限性与不稳定性.随着时间的推移,研究者开始更多地聚焦于特定的自调优问题,如物理组织自优化、自适应统计直方图、查询执行时间估算等,文献[4]对 1997 到 2007 年间数据库自调优技术的发展进行了总结.近十年来,人工智能技术,特别是机器学习、深度学习的发展,使得自然语言处理、图像识别等领域获得了很多突破.与此同时,以 DB2 的学习型查询优化器 LEO[5]为代表,学习、统计的思想和方法在数据库领域开始得到重视.人工智能赋能的数据管理新技术得到了更为广泛的关注和更深入的研究.

人工智能赋能的数据管理技术,主要指将机器学习为代表的人工智能技术中统计、学习、推理和规划等能力应用到数据库系统和数据管理中,实现减少人力开销和提高性能的目的.人工智能赋能的数据管理技术通过对查询负载、数据分布、数据库硬件特性、历史查询性能表现等进行特征抽取和建模,结合机器学习等技术,有效利用历史数据和习得模型,对数据库进行有针对性的优化.

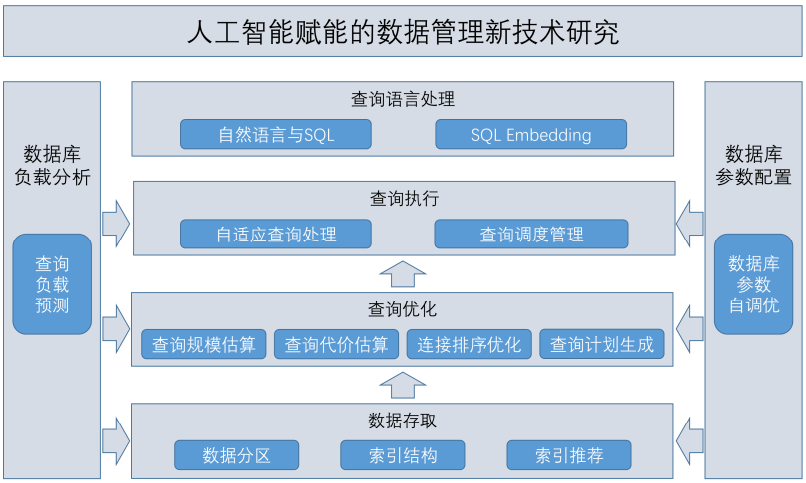

Fig.1 Roadmap of the contents of this survey
图 1 本文内容结构图

人工智能技术在各种数据库中都有应用,如对 XML 数据库进行代价估算的 COMET[6],适用于图数据库查询代价估算的 PREDIcT[7]等.本文主要关注人工智能技术在关系型数据库中的应用,以数据库系统的层次结构为划分依据,将从数据存取、查询优化、查询执行、查询语言处理、数据库运维五个方面,选取有代表性的研究成果介绍人工智能赋能的数据库的研究进展.其中数据库自动运维包括查询负载预测、数据库配置参数调优等,其结果的优劣将影响数据存取、查询优化及执行的表现;数据存取的优化包括数据分区、索引结构、索引推荐等;查询优化包括查询规模估算、查询代价估算、连接操作顺序优化、查询计划生成等;查询执行包括自适应查询处理和负载调度管理等;查询语言处理包括自然语言与 SQL 的转换、SQL Embedding 等.本文最后总结人工智能技术在数据库系统中实现方式和存在问题,并对未来的研究方向进行了展望,图 1 给出了本文内容的逻辑关系图.

1 人工智能赋能的数据库运维

数据库运维指数据库管理员为数据库正常稳定运行而进行的数据库环境部署、参数配置、性能监控、故障处理等工作.传统数据库运维依赖 DBA 的经验知识和人工操作,但由于查询负载日趋复杂且变化性增强,数据库参数配置等运维工作已超出 DBA 的能力.抽取和分析查询负载的特征、合理配置数据库参数,对数据库的整个调优过程都十分必要,人工智能技术可以根据历史负载信息,预测查询负载未来的变化趋势,自动设置和修改数据库的配置参数,提高数据库性能.

1.1 查询负载预测

查询负载特征的抽取和利用是数据库自动管理和调优的重要环节,实现查询负载预测,对数据库系统选择适当的优化方式十分重要.最初的数据库调优系统不考虑负载变化,如文献[8,9]的视图物化、索引推荐都是在离线状态进行.而对目标应用程序的工作负载进行正确的预测,可以帮助数据库实现更加有效的调优.文献[10] 提出的在线索引推荐系统 COLT 考虑了变化迅速的工作负载,但是它并没有考虑负载变化的周期性,文献[11]提出利用负载自身的周期性来减少系统工作量的思路,并针对负载变化的大小来对负载监控系统进行控制,文献[12]与它思路相似,结合数据库负载变化的规律来对 DBMS 未来的工作量进行预测,用文献[13]中的特征提取方法,从查询语句中提取特征,利用分类算法根据特征来做负载分类,将结果存到负载池中并构建一个 N-gram 模型用来对当前的负载进行类型匹配.文中的系统还会对负载的周期性进行分析,最后根据当前负载的类型和周期性给出未来负载的预测.

先前的预测技术大多基于资源利用率进行建模,当数据库的物理资源发生变化时,资源利用率也会随之改变,这样就会使预测模型失效.文献[14]中的 QueryBot5000 系统利用各种查询的到达率(Arrivall Rate)做聚类,然后根据每个聚类中查询的平均访问次数来训练预测模型.系统构建了一个由 KR(Kernel Regression)、LR(Linear Regression)和 RNN(Recurrent Neural Network)模型组合起来的混合模型做负载预测,这种混合模型可以区分和预测具有周期性、突变性和演化性等多种变化特征的查询负载.由于整个系统使用的数据独立于数据库硬件,所以即使 DBMS 的硬件或配置有所更改,也不需要重新建模,弥补了之前方法的不足.

1.2 数据库配置参数自调优

数据库系统具有大量的配置参数,这些参数控制了系统的内存分配、I/O 优化、备份与恢复等诸多方面,并极大地影响了数据库的性能.但是由于数据库配置参数数量众多,而且不独立(更改一项参数会影响其他参数)、不通用(一个应用程序的最优参数配置可能对另一个应用程序不是最优的)、没有标准化(两个 DBMS 对同一个参数设置使用不同的名称),所以对参数进行调优十分困难.随着数据库和应用程序的规模和复杂性的增长,手动配置 DBMS 来满足应用程序的需求已经超过了数据库管理员的能力.各大商业数据库也在极力研究各自的自调优系统,如 IBM[9]和 Oracle[15].

早期的调优研究着眼于调整某一部分的配置,如文献[16,17]主要关注缓冲池大小调整.文献[18]中提出的配置参数推荐系统仍然需要数据库管理员的参与,如需要管理员确定配置之间的依赖性、控制调优流程等.文献[19]提出的 OtterTune 是一个对 DBMS 参数进行自动配置的系统, 它将机器学习与参数调优结合起来,不再依赖于数据库管理员, 而且提出重用训练数据,来调整 DBMS 配置的观点. OtterTune 从已有调优计划中收集数据,将不同负载对应不同配置的数据库性能存放到 N 个矩阵中,利用矩阵实现负载映射;然后利用高斯模型,以一组初始化的配置作为起点,使用梯度下降的方法迭代找到局部最优,最后给出配置参数推荐.

近几年云数据库的应用更加广泛,但由于样本数量少、配置参数多等诸多问题,使得数据库参数调优变得十分困难.而且云数据库配置调优本身比 RDBMS 繁琐,这就使得数据库管理员的任务更加艰巨.文献[20]实现的 CDBtune 是一个端到端的云数据库自动调优系统,它采用试错策略,通过有限的样本学习参数设置,完成参数初始化,减小了对大量高质量样本采集的需求.而且系统采用强化学习中的奖励反馈机制,实现端到端学习,在高维连续空间中寻找最优配置,加快模型的收敛速度,提高了在线调优的效率.系统收到用户的在线调优请求后,会在大约 150 秒内从用户处收集查询工作负载,获取当前配置参数,并执行查询,得到当前性能,然后使用离线训

练的模型进行在线调优.最后,给出参数配置.如果优化过程终止,系统还会更新强化学习模型和内存池.

表 1 对比了这三种方法在数据库参数自调优上的异同.对于 iTuned 来说,它本身并没有使用机器学习有关的模型,调优效率受限,而且并不能完全脱离 DBA 的控制,需要手动部署数据库副本,这直接限制了 iTuned 的适用范围,使它不适用于负载多变且实时监控的应用;相比之下,OtterTune 用高斯模型来迭代搜索最佳配置,不光解决了 DBA 的问题,还提高了推荐速度,能够适应负载快速变化的应用,但是带来的缺点是对训练样本要求比较高,需要大量数据来训练模型;最新的成果 CDBTune 运用了与 OtterTune 相同的数据收集方式, 并在线下训练模型,将 OtterTune 的高斯模型换成了强化学习,这就使得系统减轻了对训练样本的依赖,增强了系统的适应性,能够完成对云数据库的参数配置推荐.

Table 1 The comparison of different auto-tuning systems

表 1 数据库配置参数自调优方法对比

调优系统	iTuned[18]	OtterTune[19]	CDBtune[20]
算法模型	非机器学习模型	高斯模型	强化学习模型
适用场景	负载变化不频繁	负载变化频繁	云数据库
DBA 参与	需要	不需要	不需要
特点	调优时间长	对训练样本要求高; 使用迭代方法可以在短时间内给出初步推荐结果	系统适应性强; 强化学习算法可以利用有限的样本给出最优推荐

1.3 小结

数据库参数自动调优是人工智能技术在数据库运维中的主要应用,基于人工智能技术的负载预测和配置参数调优,可以有效利用历史负载信息和数据库性能表现,提高数据库在数据存取、查询优化、查询执行等多个方面的性能.

2 人工智能赋能的数据存取

传统数据库中,通常由数据库管理员依据个人经验或是启发式规则来维护索引、进行数据分区.人工智能赋能的数据存取优化则以历史查询负载和数据分布特点作为输入,利用机器学习和数据挖掘技术,合理地进行数据分区、选择在哪些列上建立索引或是以机器学习模型替代传统的索引结构.人工智能赋能的数据存取可以使数据存取的优化动态性更强,分区和索引对数据、负载和硬件更具适应性,以达到提高数据检索、存取速度的目的.

2.1 数据分区优化

数据分区(Partitioning)是数据库物理设计的重要部分.数据分区主要包括水平分区(Horizontal Partitioning)和垂直分区(Vertical Partitioning),通过数据分区,表、索引或者物化视图被划分为元组和属性的子集,对数据库的性能和可管理性的提升有重要意义.

在过去的几十年里,人们对此问题进行了大量的研究,如文献[21]关注单机的数据分区,文献[22,23]对分布式数据分解进行研究.以往的研究主要根据数据库模式(Schema)、数据分布和负载的特征生成范围(Range)或哈希(Hash)分区,然后使用启发式方法和代价模型对分区进行评估.

文献[24]提出了适用于分布式 OLTP 数据库的 Schism 系统.Schism 是一个负载驱动、基于图分割算法的分区系统.其分区过程包括两阶段,首先,Schism 创建一个数据负载关系图,图中的节点是采样出来的数据库中的元组或元组的集合,图上的边表示相连两点所代表的元组被同一个事务访问,由此,Schism 将数据分区问题转化为图分割问题.用 METIS 算法[25]将图分割为 K 份,表示相对应的数据被分区到 K 台机器,以实现减少跨节点事务、保持数据均匀分布的目的.由于内存大小的限制,Schism 只能在采样的结果上进行分区,为了能够实现所有数据的分区,在第二阶段,Schism 通过决策树模型来学习和解释分区策略.决策树以已分区的采样数据中频繁列元组的属性值为输入,以其分区结果为输出,通过决策树,可以将新元组根据其属性值进行分区结果的预测.最

后,Schism 将基于机器学习的分区策略与传统的基于哈希的方法等进行对比,选择效果更好的作为分区依据.但是 Schism 系统并不支持独立事务处理[26],并且没有考虑分区之间数据的依赖关系.文献[27]对 Schism 进行了改进,文章给负载关系图的边赋予了权重,如为具有数据依赖关系的对象的边分配了最高的权重,为独立事务分配尽可能小的权重等,通过对带权图进行分割,实现数据分区.

文献[28]在列存储数据库上尝试使用强化学习的方法进行数据分块和数据分布方式的选择,针对一组特定的查询负载和数据,使用完成一组查询或查询样本的估计时间作为代价函数,通过合并、拆分等动作,使用基于 Q-Learning 的强化学习算法探索学习适合该组负载和数据的分块方式.但是文章在状态表示和代价函数的选择的合理性上只进行了初步的验证,也没有在真实数据库上进行实验,在可扩展性和适用性方面仍然存在问题.文献[29]将强化学习的思想用于分布式数据库的分区问题,与前文不同,本文除了用负载的估计执行时间作为代价函数进行离线学习阶段,还设置了在线学习阶段,根据离线学习的模型进行数据分区,以数据分区后负载的实际执行时间作为代价函数,对强化学习模型进行调优.在线学习的使用避免了强化学习模型受限于传统代价估算模型的质量,同时可以增强模型处理新负载的能力.

Table 2 The comparison of different partitioning methods

表 2 分区方法对比

名称	Schism[24]	Automated Data Partitioning[27]	GridFormation[28]	Partitioning Advisor[29]
算法模型	图分割算法	图分割算法	强化学习	强化学习
适用场景	OLTP 数据库	OLTP 数据库	OLTP 数据库	OLAP 数据库
是否分布式	是	是	否	是
分区方式	水平分区	水平分区	垂直分区	垂直分区
特点	样本上的图分割、决策树模型一般化	根据属性的依赖关系赋予边的权重	使用优化器估算的代价作为代价函数	在线学习阶段使用分区后真实代价做代价函数

表 2 对比了近年来数据库分区的新方法,Schism 与 ADP 通过将问题转换为图分割问题,再用机器学习模型从学习图分割中学习分区策略,ADP 在 Schism 的基础上为表示数据依赖关系的边赋予了不同的权重,使之变为带权图的图分割问题,获得了比 Schism 更好的结果.GridFormation 和 Partitioning Advisor 则将强化学习的思想应用于分区,GridFormation 使用每次分区后估算的查询性能表现做为代价函数,使其训练开销小于使用真实代价做为代价函数的 Partitioning Advisor,但是其结果的可靠性受到代价估算模型的限制.

2.2 索引结构

索引可以加快数据库中数据的检索速度,但是建立和维护常见的索引结构通常需要消耗大量的时间和空间资源.文献[30]指出,索引可以看作一种用来定位数据在存储中位置的模型,而这种模型完全可以由机器学习的技术来构建,即 Learned Index.文章解释了如何在了解数据分布的情况下,用 Learned Index 进行范围索引、点索引和存在性索引.传统范围索引的代表是 B 树,而将索引视为模型的时候,键(Key)为输入,对应键的记录的位置作为预测输出,保证它的最大误差小于一个内存页即可,B 树中的页大小(PageSize)即相当于机器学习模型中的最小最大误差(Min-/Max-error).文章使用了递归模型(Recursive Model)架构,在有序数据上用神经网络替代 B 树进行范围索引.模型对有序数组内键的近似累计分布函数(CDF)进行建模,预测数据的位置.在有序数据上 B 树检索的时间复杂度仍然 O(logn),而机器学习模型可以将其时间复杂度变为常数.点查询的代表是哈希算法,哈希的查询时间复杂度较低,所以用机器学习模型来代替哈希主要是为了减少冲突,从而减少键的存储空间.文章对点查询的模型不再限于有序数据,同样使用递归网络结构,学习键的经验累积分布函数.判断某个键在一个特定数据集中是否存在的代表方法是布隆过滤器,但是布隆过滤器有一定的误判率,如果要提升精度减少误判率就势必要增加 bitmap 的大小.对于布隆过滤器来说,存在的键和不存在的键在自己的函数内冲突越多越好,这样可以用更小的 bitmap 表示更多的键.文章通过训练一个神经网络,使得损失函数最小,来满足假阴性为 0 的条件,然后再创建一个溢出的布隆过滤器,根据阈值学习一个模型,当输出结果大于等于阈值时,认为这个键在数据集中,当小于阈值时,则去溢出的布隆过滤器检查.该模型可以提高空间利用率,而且精度越高,内存占用越少.

2.3 索引推荐

索引选择在 70 年代就开始被研究,如文献[31,32]等,他们的研究大多停留在单个关系的单索引选择上;文献[33,34]则讨论了针对多关系上的单索引选择问题,真正第一次实现 RDBMS 的索引推荐的是 1997 年的文献[35],它在 Microsoft SQL Server 上部署了完整的端到端的索引推荐系统.该系统利用了文献[33]所提及的"原子配置"概念,用"原子配置"的查询开销来估计整个索引配置的查询开销,这就极大地减少了查询优化器执行工作负载的时间.文中索引推荐算法的核心思想是:如果一个索引不在某个查询的最优索引配置中,那它就不可能在整个工作负载的最优索引配置中.系统从单个查询的单个索引开始迭代,寻找整个负载下的最佳索引配置.这些早期的索引推荐都是在离线状态下进行的,不会根据负载变化重新选择数据库的索引,随着数据库负载的改变,先前推荐的最佳索引极有可能会失去优势.

文献[36,37]是早期与在线索引配置相关的研究,它们持续监控工作负载,选出候选索引,分析索引之间的优劣后给出推荐.它们都用存在性假设"What-if"来判断是否在某一列上构建索引,但是对"What-if"的控制有所不足,不管系统性能是否有必要进行调优,都以同样的强度来做索引配置,这显然是不可取的.COLT[10]系统在这方面有所完善,它利用文献[38]的预测函数来评估是否在某一列上构建索引,给出一组衡量数据库物理设计优劣的指标,并持续监控工作负载,每执行 N 个查询,就重新考虑索引配置是否需要更新,如果系统性能好,在线调优的强度就会降低,而如果当前工作负载变化较大,在线调优的强度就会增加.在线索引配置主要适用于查询模式变化不太频繁的工作负载.如果工作负载变化很快的话,先前研究的在线索引推荐性能就会明显下滑.文献[39]自调优系统中的自适应索引解决了离线索引在线索引的局限性,它可以增量式地构建优化索引来适应工作负载的变化.

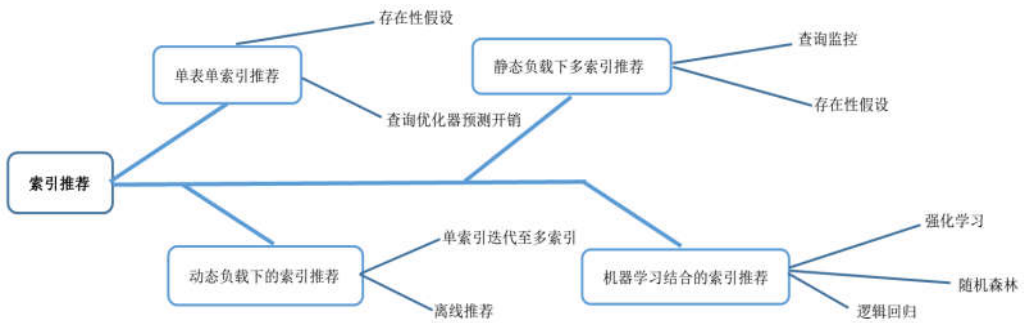

Fig.2 The development process of index recommendation

图 2 索引推荐发展过程

在后来的研究中,数据挖掘与机器学习也被应用到了索引推荐中.文献[40]用频繁项集挖掘的方法来选择推荐索引的候选集,然后从候选集中选择使查询执行时间最短的索引集作为推荐.2018 年的 NoDBA 系统[41]又提供了一个用强化学习进行索引选择的思路.将工作负载和当前配置作为神经网络的输入;将在特定的列上创建索引定义为可执行操作.由完全没有索引为起点,进行强化学习,最后给出推荐的索引配置;其中奖励函数用构建某一列索引带来的性能提升来表示.整个系统按照强化学习的流程运行,接收索引和负载,由初始状态开始迭代,根据奖励函数判断某一特定列上是否应该构建索引,当索引数量达到上限或者系统性能到达阈值时停止迭代.该实验证明了为这样的任务训练神经网络的可行性.这些索引推荐大多基于查询优化器的成本估计,给 DBMS 造成不小的开销,而且存在性能倒退的可能性.文献[42]与文献[41]一样运用机器学习的思路,从比较同一查询的不同索引配置的执行成本这一个关键步骤入手,将是否建立索引转换为一个二分类问题,减小了查询优化器成本估计不准确造成的影响.它把查询计划作为分类任务的输入,将比较两个查询计划的优劣,转换为比较两个计划代价的问题.文章使用线性模型和树模型来训练分类器,线性模型用逻辑回归,树模型用随机森林和梯度

提升树.文章将分类器集成到数据库的索引推荐系统中,通过大量工作负载测试表明,在一对查询计划的比较中,错误可以减少为原来的五分之一,这就极大地减少了索引推荐时的性能倒退的可能性.

图 2 展示了索引推荐的发展过程.单表单索引推荐的提出启发了人们对索引自动构建的思考,但是由于单表的限制使得其使用范围极其狭窄;在单索引推荐基础上衍生的多索引推荐,适用范围更加广泛,它可以完成负载变化不是很频繁的应用的索引推荐工作,但是由于网络的普及以及用户数量的爆炸式增长,更多应用的负载都是不断变化的,所以研究人员把着眼点放到了动态负载下的索引推荐研究上,来解决负载快速变化的问题;在研究动态负载索引推荐的同时,机器学习的技术逐渐成熟,它在大规模数据的高效运用以及推荐预测方面的高准确率引起了科研人员的注意,随后诸多机器学习的算法被运用到了索引推荐的研究中,机器学习的加入使得索引推荐不仅能够适应负载的快速变化,更能将大量的用户数据利用起来提高索引推荐的效果.

2.4 小结

人工智能技术特别是神经网络、强化学习的方法和思想用于数据库的数据存取,可以帮助数据库管理员更好的进行数据分区和索引推荐.而基于网络模型的新型索引结构,为多维数据索引也提供了新的方法.

3 人工智能赋能的查询优化

查询优化是数据库系统最重要的任务之一,大多数数据库系统使用基于代价的模型,通过枚举查询执行计划、估算每个枚举的开销来选择开销最小的执行计划.查询性能非常依赖优化器的质量,人工智能赋能的查询优化,可以根据每一次查询估算和实际执行的表现来不断优化查询规模估算和查询代价估算的质量.在各类查询负载中,连接操作非常常见且代价昂贵,依托强化学习等技术的连接排序优化也得到了广泛的研究.

3.1 查询规模估算

数据库最佳查询执行计划的选择需要对备选计划的代价进行准确的估计,而查询规模估算是查询代价估算最重要的因素之一[43].传统的查询规模估算可以分为四种方式,以统计直方图[44,45]为代表的无参数方法,用概率分布函数预测的有参数方法[46,47],曲线拟合(Curve Fitting)[48,49]方法和采样[50]的方法.在应用中,这些方法并不是割裂的,也有很多混合方法,如文献[51]等.

传统的估算方法是静态的,即不会根据历史估算准确性的表现来调整估算过程,如在曲线拟合的方法中,用于拟合数据分布的多项式函数的系数是固定不变的.在 1994 年,文献[52]最早将自适应的学习方法应用到查询规模估算中,他们在每次以多项式函数拟合查询规模后,用循环最小平方差(Recursive Least-square-error)衡量真实查询规模与预测查询规模的误差,并使用其作为代价函数用来更新多项式的系数,以此不断优化函数对查询规模的估算准确率.传统方法很少或者几乎不会对拟合函数的系数进行更新操作,而这种方法实现了系数的动态调整,因此也获得了更高的预测准确率.

在核密度函数估算的方法中,核函数(Kernal Function)的带宽(Bandwidth)参数对估算结果影响巨大[53],如何选择一个合适的带宽参数是一个极具挑战性的问题.文献[54]针对这个问题提出了新的解决方案:他们选择了连续可微的高斯核函数,在使用传统的斯科特法则(Scott's Rule)初始化带宽参数后,使用梯度下降的机器学习算法,对每次查询执行,将估算的结果和真实结果的积分均方误差作为损失函数,对带宽进行优化,并且将 GPU 的并行运算能力应用到调优过程中,获得了优于传统核密度函数方法和当前最好的统计直方图方法的估算质量.在此基础上,文献[55]针对连接操作,使用高斯核函数和线性逼近约束优化(Constrained Optimization by Linear Approximation)对连接操作的查询结果进行估算,同时面对数据库的增删改和不同的查询负载,动态的调整抽样结果和核函数带宽参数,保证了估算结果的准确性.

一些文献不再对数据分布情况进行建模和估计,而是利用查询语句和查询计划中的信息,使用机器学习的方法预测查询规模.文献[5]试图在预测的查询规模与真实规模之间直接寻找关系,文章引入了一个反馈循环,通过在预测的查询规模与真实规模之间建立线性模型,使用具有相同谓词的历史查询的实际值来调整估计值,实现使优化器从错误中学习的目的.这种方法虽然实现了学习修正查询规模预测的目的,但是并没有学习数据的

分布情况,由于查询规模预测错误来源的复杂性,简单的线性模型通常不能完全纠正查询规模预测错误.文献[56]提出了一种黑箱方法,这种方法不再为查询计划的每一层进行查询规模估算,而是将历史查询语句聚类为查询模板,使用分类与回归、模型树、局部加权回归等机器学习技术学习每个查询模板中查询的属性、常量、操作符和用户定义函数等参数及其基数分布.这种方法同样不再依赖数据分布,也抛弃了传统的自底向上逐层估算的方法,可以减少分布式数据库环境中的网络传输代价.文献[57]提出了适用于云服务场景下的查询规模估算学习优化方法,由于云服务中负载具有重复或部分重叠的特征,因此对历史负载中具有相同查询类型但涉及不同参数和数据的子查询建立查询模板,更利于进行特征提取和泊松回归模型建立.这三种方法都以查询模板为单位进行学习建模,当处理历史数据中未出现过的查询负载时,预测准确性会受到很大的限制.

神经网络同样被用于估算查询规模,早在1998年,文献[58]就将神经网络应用于用户定义数据类型和用户定义函数的查询规模估算.由于用户定义的数据类型通常是多维、非数值型的数据,因此传统的统计直方图等方法对此类数据的适用性差.文章提出了使用迭代解析元数据的方法,将解析出的元数据作为神经网络输入的特征向量,用真实的查询规模来训练网络,实现对用户定义数据和函数的查询规模估算.这种方法虽然对较为复杂的用户定义数据类型提出了查询规模估算的学习方法,但是仍然限制新数据类型的元数据必须是数值型数据.

由于数据的分布并不一定是均匀的,神经网络中普通的激活函数不能准确地拟合查询规模的变化情况.文献[59]将SQL语句中数值型数据的范围查询阈值作为特征向量,使用阶跃激活函数[60]搭建了增强的神经网络模型,相较于文献[58]的方法,在倾斜数据集上的预测获得了更高的准确率.文献[61]除了使用查询范围的阈值做特征向量,还利用了数据库中每个表的统计直方图信息.通过将每个表一维统计直方图的AVI(Attribute Value Independent)、EBO(Exponential Backoff)和MinSel(Minimum Selectivity)信息加入特征,使特征可以更直接的反映数据的分布情况,提高了文章中神经网络、随机森林和梯度提升树模型的准确率,同时也增强了面对数据库中数据变化时模型的健壮性.文献[59,61]关注的是查询执行计划叶节点即表扫描操作的查询规模,连接操作同样是查询规模估算中的关键问题,文献[62,63]提取了查询语句的表属性、连接操作属性和查询谓词三部分特征,作为一种新的神经网络模型——多集合卷积网络(Multi-set Convolutional Network)的输入,用于多路连接的查询规模估算.

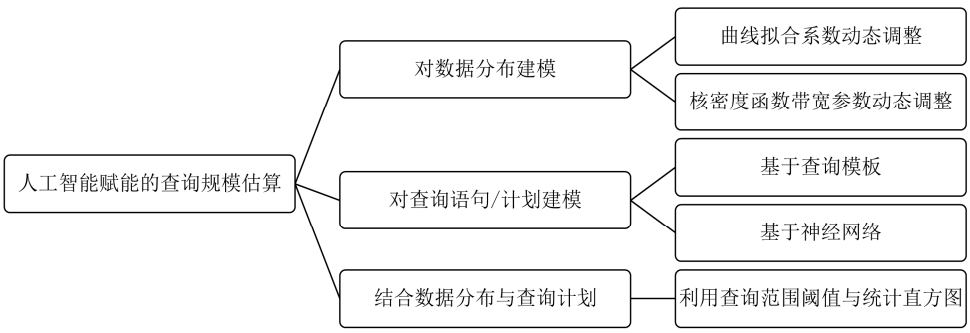

Fig.3 Classification of cardinality estimation methods based on artificial intelligence

图3 基于人工智能的查询规模估算方法分类

在机器学习或深度学习的应用中,特征工程是非常重要的环节,特征选择的优劣直接影响学习结果的好坏.文献[56-59,61-63]中模型特征向量的选择是一个人力密集型的过程,为了能够生成数据稠密且信息丰富的特征向量,文献[64]提出了根据查询计划,自动生成查询操作符特征向量的新方法.文章利用自然语言处理中词嵌入(Word Embedding)的思想,搭建一个沙漏形神经网络结构,以查询计划中父节点的信息为输入来预测其子节点的信息.在该网络经过训练后,截取网络中间层的结果作为该父节点查询操作的特征向量,将这个特征向量用于查询规模的预测等多个任务.通过这种方法获得的特征向量,可以更充分的包含操作符的语义和结构信息,同时

可以用于多个后续的学习任务,通用性更广泛.

人工智能赋能的查询规模估算分类如图 3 所示,这些方法有效地利用了历史查询的结果信息,将实际的查询规模用于估算模型的调整.同时,这些方法可以放宽传统估算方法中属性独立、连接均匀等假设的限制,较为准确地预测多属性查询和多表连接操作的查询规模.其中对查询语句或计划建模的方法不再直接考虑数据分布情况,可以减少估算器对底层数据的访问.而文献[61]由于结合了数据分布与查询计划的信息,其估算结果不仅优于传统方法,也超过了文献[54]中 KDE 的学习方法.

3.2 查询代价估算

查询代价估算,通常指预测估算某个查询计划在给定硬件和系统参数配置下的执行时间和资源消耗.查询代价估算除了可以指导查询计划的选择,对查询调度、数据库资源管理等也有重要意义.但随着数据库变得越来越复杂,负载类型越来越多样,量化 CPU 和 I/O 的代价变得越来越困难,传统的代价估算模型在估算准确性上的表现不尽人意.除了提高查询规模估算准确率以优化查询代价估算,一些工作也将人工智能技术直接用于查询代价估算,其分类如图 4 所示.

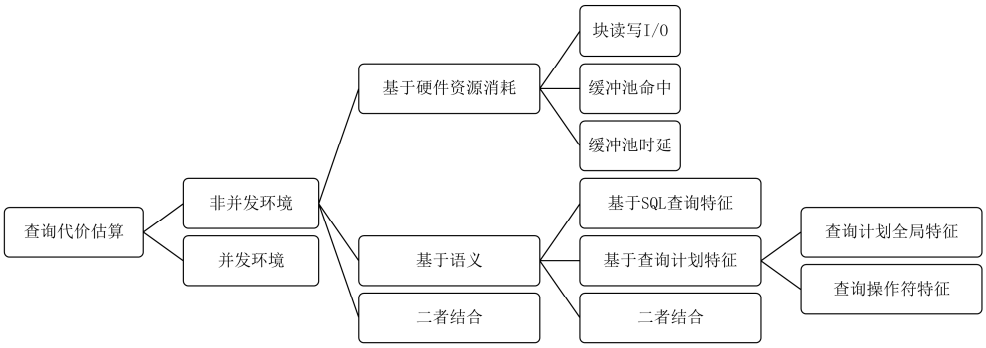

Fig.4 Classification of cost estimation methods based on artificial intelligence

图 4 基于人工智能的查询代价估算方法分类

3.2.1 单查询代价估算

文献[65]用 KCCA(Kernel Canonical Correlation Analysis)机器学习模型替代了传统查询代价估算中的代价模型,选取 SQL 查询中选择谓词、等值连接操作的数量特征和查询执行计划中各操作符的数量及其查询规模之和等特征作为输入的特征向量,建立模型,学习和预测查询执行时间、访问元组数目、结果元组数目、磁盘 I/O、消息计数(Message Count)、消息字节(Message Bytes)等 6 种性能表现.文献[65]仅提取了查询计划级别的特征作为模型的输入,文献[66-68]则考虑了查询计划中每一步原子操作对查询代价的影响.文献[66]用支持向量机和线性回归模型替代传统的代价模型,在查询计划整体和原子操作两种级别分别提取特征进行训练估算.查询计划整体提取了查询计划估计时间、查询结果估计规模、查询计划中原子操作的数目等特征;原子操作则对查询计划的每一步操作提取了操作执行估计时间、单步操作估计规模、该操作左右子树的相关信息等作为特征向量.结果显示粗粒度、查询计划级别的模型预测准确率更高,但对未出现或者动态变化的负载支持差,而细粒度、原子操作级别的预测模型通常有着更好的扩展性.文献[67]在前文的基础上提出了一种混合模型:在尽可能多的使用原子操作级别的预测模型情况下,对查询子计划使用计划整体级别的预测.这种混合模型避免了由于操作级别预测引起的误差经过查询计划树自底向上逐层累积后导致的误差扩大,在增强了对新负载的适应性的同时,获得了较高的预测准确性.文献[68]同样在原子操作级别进行查询代价的估算,文章对每类原子操作提取了共性特征(如操作输出的元组数目等)和特有特征(如表扫描操作中输入的数据页数、排序操作中排序列

的数目等),通过为每个原子操作训练一个基于随机梯度提升的回归树模型(Boosted Regression Trees),预测操作的执行时间等代价信息.文章还将缩放函数(Scaling Funcions)与回归树组合成联合模型,以增强模型对新负载和数据分布的适应性.

文献[66-68]以操作为单位预测代价,再根据查询计划自底向上的累加各操作以得到查询的总代价.文献[69]则考虑了查询计划的树结构中,子节点操作对父节点操作的影响.文章对每类操作训练一个神经网络模型,该网络除了输出当前操作的执行时间,还输出一个向量,这个向量和该操作父节点的特征一同作为父节点神经网络的输入.这种与树型结构相匹配的模型,可以更有效的利用查询计划结构的信息,获得了比[67]更优的预测结果.

3.2.2 并发查询代价估算

文献[65-69]对查询执行时间的预测都将查询执行限制为单查询负载的情况,然而在实际的数据库应用场景中,往往面对的是多用户、多查询负载甚至是多节点的分布式环境.并发查询一方面有利于提高硬件的资源利用率和聚合吞吐量,但与此同时,并发查询也带来了很多挑战,查询执行时间预测也变得更加困难.并发的查询之间如何影响彼此的执行进度是造成难点的主要原因,由于多个查询竞争硬件资源,它们之间的关系可能是促进的、抑制的或者互无影响.比如当两个查询涉及到同一个表的扫描操作时,其中一个查询可以利用另一个查询缓存的数据,从而获得查询速度的提升,而当两个查询访问完全不同的数据时,可能由于 I/O 限制,两个查询的执行速度都会变慢.因此在并发条件下预测查询的执行时间就变得更加困难.但准确的预测并发执行时间却有很大的意义,如可以进行合理的并发调度,减少负载总体执行时间,为用户提供更好的查询服务等.文献[70]最早使用机器学习方法预测并发环境下查询执行时间,文章提出了一种二叉树模型 PQR,用以预测在负载并发的情况下,某个查询计划执行时间的范围区间.模型的输入不仅考虑了查询执行计划的特征,还将当时系统负载的 MPL(Multi-Programming Level)特征作为输入.二叉树每一层的结点包含一个二分类的分类器,可以根据输入的特征向量预测其更为细分的查询执行时间范围,经过逐层计算后得到该查询的执行时间范围.

文献[70]使用 MPL 来衡量系统负载,文献[71]则使用了更直接且更细粒度的性能指标 BAL(Buffer Access Latency),BAL 是逻辑 I/O 操作的平均时延,即当查询引擎提出一个数据块访问请求,到符合该请求的数据块被放回到查询引擎的平均执行时间.BAL 反映了查询的生命周期内数据存取的平均表现,因此比缓冲池命中数目和物理块 I/O 数目更能准确地表现复杂查询与系统和数据的交互.通过实验对比,在单查询负载的情况下,BAL 比缓冲池命中和物理块 I/O 数目更能准确地估计查询的执行时间.对于负载并发的环境,文章对查询时间的估算以查询模板为单位,将负载分为主查询(Primary)和从查询(Complement),通过 B2cB 模型和 B2L 模型估算负载并发中主查询的执行时间.在数据收集阶段,收集每个查询模板在非并发环境下的 BAL 和每个模板在与其余各模板并发执行时 BAL 的变化量.同时,为了充分获得各类负载并发的训练数据,文章使用了 Latin Hypercube Sampling(LHS)的采样方法生成负载并发的数据集.B2cB 和 B2L 都是线性回归模型,其中,B2cB 模型以主查询在非并发环境下 BAL 值、主查询与各从查询并发环境下 BAL 变化量和各从查询之间并发环境下 BAL 变化量为参数,建模估算主查询在并发环境下的 BAL,B2L 模型以主查询 BAL 为输入,建模估算主查询在并发环境下的执行时间.文章比较了 Just-in-Time 和 Queue Modeler 两种并发方式下新查询的执行时间及其对当前负载的影响.JIT 方式即不管当前负载情况如何,新查询提交后立即执行,Queue Modeler 则是限制了并发负载的数目,并将新查询放入队列,当并发查询数目小于限制时才开始执行.与此类似,文献[72]改进了 LHS 采样算法,使采样结果能够充分包含各种查询并发的情况,并且应用混合高斯过程模型对一组查询负载的执行时间进行建模预测;文献[73]设计了新的采样方法 CDR(Corner, Diagonal, and Random),并且比较了线性模型和回归树模型以 NRO(Normalized Run-time Overhead)为特征时,对并发负载执行时间的预测结果.

以上的方法都是以整个查询为单位考虑并发的情况,文献[74]将负载的并发情况进行了更细粒度的分析.文章以查询计划中操作符是否需要等待全部输入才能进行下一步的输出为标准,将操作符分为阻塞(Blocking Operator)和非阻塞(Nonblocking Operator)两类,如排序操作需要获得其子节点的所有中间结果之后才能进行排序结果的输出,因此是阻塞类操作符.对每个查询计划,根据操作符类型,将查询计划的执行过程分割为多个子流程(Pipeline),由此将查询之间的并发变为查询子流程之间的并发,使得并发负载分析的粒度更细化.文章选择

了并发情况下描述各查询子流程 CPU 占用和 I/O 情况的九种特征作为特征向量,用线性回归模型和回归树模型对并发查询子流程的时间进行了预测.文献[70-74]都仅利用了资源消耗的统计信息进行并发执行时间预测,并且假定了静态的工作负载场景,静态意味着将要运行的查询所属的查询模板是固定的且已知的,但实际情况通常并非如此.文献[75]提出了 Contender 模型,增强了模型对未知负载在并发情况下执行时间的预测准确率.文章除了考虑并发负载下的资源消耗情况,还将并发查询对同一个表的扫描操作数量等并发负载的语义信息作为模型的特征.当负载中出现新的查询模板时,Contender 先单独执行新查询并收集其 I/O 时间占比及最大中间结果信息,通过对已知负载的 I/O 时间占比和最大中间结果建立 KNN 模型,用模型估算新负载在并发情况下的资源消耗情况,再用线性模型预测其在并发情况下的执行时间.

3.3 连接排序优化

连接操作(Join)是数据库中非常常见的查询类型,大多数的连接操作涉及的表不会超过 20 个,但实际应用中仍会有数百个表的连接操作查询[76].连接排序是一个 NP-hard 问题[77],面对多表连接排序,现有的方法通常使用动态规划[78,79]、贪心算法[80,81]等启发式算法来优化连接排序,但这些方法非常依赖查询中间结果的规模估算[82],并且为了将优化时间控制在合理的范围内,这些方法牺牲了很大的优化效果,而且在面对超过 1000 个关系的连接操作时,这些优化方法几乎都无法给出合理的优化顺序[83].

文献[84]将连接排序问题看作马尔可夫决策过程(MDP),提出了一种基于深度强化学习的连接排序优化方法.该方法将每个 SQL 查询表示为查询图(Query Graph),关系 R 和 S 的连接操作 c=(R,S)将查询图中的点 R 和 S 合并为一个新点(R+S),原来和 R、S 连接的边则指向新点(R+S).这样,连接排序问题转化为一个 MDP 过程,查询图 G 是 MDP 的状态(State),一次连接操作 c 是 MDP 的动作(Action),每次连接操作的估算代价 J 的倒数作为奖励(Reward).在将 G 和 c 进行特征化表示并收集一定训练样本后,使用基于 Q-learning 的深度强化学习方法,训练神经网络模型,来得到适合当前数据分布和查询负载的连接顺序.文献[85]提出了 ReJOIN,使用的策略梯度下降的优化策略和二叉树的特征表示方式,基于深度强化学习方法对连接排序进行优化.这些基于强化学习的优化方法不仅在优化的性能上接近甚至超越传统方法,并且可以处理传统方法难以优化的大规模的连接查询.

3.4 查询计划生成

传统数据库在对查询负载进行语法分析后,生成逻辑查询计划,根据代价估算结果,评估和优化逻辑计划,最终生成物理执行计划.这个过程是由复杂的启发式算法驱动,不仅需要大量的调优、维护工作,并且无法有效利用历史查询的信息来帮助生成更优的查询计划.除了利用人工智能技术优化查询规模估算、查询代价估算和连接排序,还有一些工作直接对查询计划生成展开.

文献[86]在学习到子查询状态和操作符动作的特征表示之后,除了应用到查询规模估算的优化,还用强化学习的方式,以每一步动作之后的查询规模作为代价,优化逻辑执行计划.本文选取查询规模做代价函数,并不能保证优化出的执行计划是一个全局最优计划,并且状态-动作空间非常大,因此文章中 Q-learnig 算法的适用性较差,但是利用强化学习生成查询计划的思路为后续研究提供了新方向.文献[87]也提出用强化学习生成查询计划的思路,文章采用了两种强化学习的新方法:模仿学习(Learning from Demonstration)和代价模型引导(Cost Model Bootstrapping)来缓解初始学习效果差、训练代价高、状态-动作空间大等问题.在模型训练初始阶段,通过模仿传统数据库中优化器的优化策略、以传统代价模型作为代价函数,经过一段时间训练后,微调模型的优化策略和代价函数,实现端到端的基于强化学习的查询优化器.

文献[88]提出了一个端到端的基于机器学习的查询计划优化器 NEO.NEO 可以完成连接排序、物理操作符选择、索引选择等查询优化内容,从而完全替代传统的优化器.NEO 首先收集传统优化器优化后的查询计划及其执行时间,使用压缩的邻接矩阵提取查询计划中表连接的信息,使用 One-hot 编码、选择率改进的 One-hot 编码和词嵌入三种方式提取表扫描操作的信息.NEO 根据查询计划的树形结构,构建与查询树结构相同的查询计划特征树,并且使用树形的卷积神经网络[89]对查询计划和执行时间建模,实现了根据查询计划预测查询执行时间的目的.由于候选查询计划空间巨大,NEO 构建一个小顶堆,用最佳优先搜索策略(Best-first Search)来生成最

优执行计划,这种方法也可以视为强化学习思想的应用.

3.5 小结

基于代价模型和启发式规则的传统查询优化器在面对复杂的查询负载和数据分布时,性能会大打折扣.将学习的思想应用到查询优化,在查询优化器与查询结果及其性能表现之间建立闭环回路,使得优化器从查询的性能表现中不断提高查询规模和查询代价的估算准确性,从而制定更优的执行计划.图 5 展示了基于人工智能技术的查询优化的一般过程,除了对优化器的规模估算和代价估算组件进行学习调优,利用强化学习生成查询执行计划的方法也为查询优化提供了端到端的解决方案.

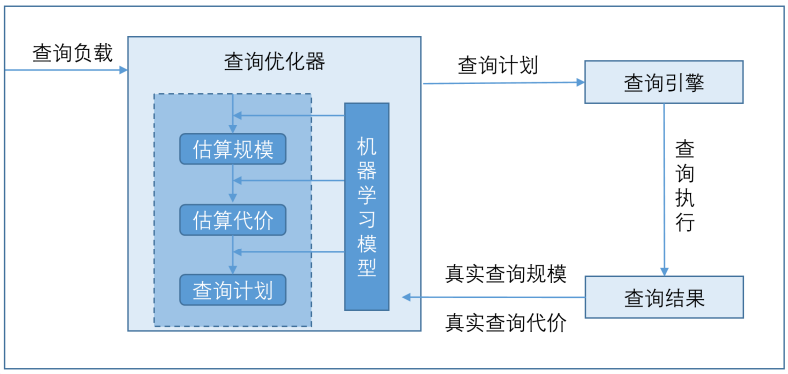

Fig.5 Process of query optimization using artificial intelligence

图 5 基于人工智能的查询优化的一般过程

4 人工智能赋能的查询执行

在查询执行时,复杂的并发负载和数据分布可能使查询优化器估算的代价误差很大,导致计划不能达到最优,甚至陷入糟糕的查询执行计划中,同时如何调度执行并发的负载,关系到负载的总体执行时间,最终影响到系统性能和用户体验.自适应查询处理可以在查询执行的过程中继续修改优化查询计划,查询调度可以调度查询的执行顺序、资源配置等,一些研究将人工智能技术应用到查询执行过程,帮助系统获得更好的性能表现.

4.1 自适应查询处理

自适应查询处理(Adaptive Query Processing)[90,91]是查询优化的另一种手段,由于查询优化中的查询规模估算和查询代价估算的误差可能导致查询计划陷入局部最优,自适应查询处理技术可以在查询的执行过程中继续修改查询执行计划,以达到查询优化的目的.一些自适应查询处理技术在优化过程中,仍然依赖于查询规划的启发式规则和代价估算模型[92],新的方法开始将学习的思想应用到自适应查询处理中.

文献[93]设计了 Eddy,Eddy 是位于数据库关系表与关系操作符之间的路由,Eddy 可以通过动态的选择元组将要经过的操作符的顺序,以此实现自适应的查询处理.文章定义了对称时刻(Momentsof Symmetry),在对称时刻,操作符可以进行重排序.以嵌套循环连接为例,当完成一次内循环时即进入对称时刻.具体来说,当完成外表中一个或一组元组与内表所有元组的扫描连接时,记录下外表此时指针位置,交换内外表连接顺序,此时的内表为原外表指针与结尾之间的数据,查询可以继续进行且不受到连接顺序交换的影响.Eddy 中元组可以进行操作顺序的选择,因此 Eddy 的路由策略将会决定系统的效率,进而影响查询处理的性能.Eddy 使用了文献[94]中彩票调度的学习方法,不断根据元组执行操作符后的代价学习操作符的执行顺序,从而获得良好的整体效率.文献[95]提出了 SkinnerDB,与 Eddy 类似,SkinnerDB 关注的是查询执行过程中的连接排序优化问题.SkinnerDB 在查询的执行过程中使用强化学习避免初始查询计划局部最优的问题,SkinnerDB 将查询的执行过程分割为多个时间片段,在这些时间片段中利用每个片段的部分查询规模,使用强化学习,以不同的顺序执行连接操作.这种在查

询执行中进行学习优化的方式,可以保证学习的知识是可以直接用于当前查询的,避免了未来查询负载与历史学习记录差别较大带来的问题.与 Eddy 相比,SkinnerDB 可以提供预期执行时间和最优执行时间之间关系的保证,而即使 Eddy 的表连接产生了不成比例的开销,Eddy 也不会丢弃中间结果.

文献[96]提出了微型自适应(Micro Adaptivity)框架,这个框架基于向量化查询引擎[97],查询操作符每次操作是在一组元组上进行而非一条元组.在向量化查询引擎中,每个关系操作符都由原语函数(Primitive Function)构成,并且每个原语函数都可以有多种实现方式.这种操作实现的多样性为查询执行提供了多种可选择的方式,框架在查询执行期间使用强化学习的方法自动在多个操作符实现之间进行选择,而不需要任何显式优化规则.因为系统只在最初查询计划生成时使用代价估算模型,而在查询执行过程中使用实际的查询执行的代价用于强化学习的奖励函数,因此系统的鲁棒性得到了提升.但是这种方法仍然有一些限制,首先,这种方法仅限于使用向量化执行模型,而向量化的方法需要设置超参数,并且这种方法没有研究上下文调优,文献[98]对这种方法进行了改进.文章提出了新的自适应查询处理框架 Cuttlefish,Cuttlefish 基于汤普森采样(Thompson Sampling)的概率匹配技术,可以以不同的运行特性对操作符调优,而不需要开发人员手动调整任何参数.当开发人员可以收集关于输入数据和工作复杂的相关上下文时,Cuttlefish 的调优器通过在线学习模型,为工作负载的每个部分选择最优物理操作符,而不是使用单个平均最佳的物理操作符.相较于文献[96],Cuttlefish 不再局限于对关系操作符的支持,将自适应查询处理能力扩展到了用户定义函数、卷积等非关系操作符.

4.2 查询调度

传统数据库对负载并发大多使用简单的先来先服务策略,然而,为特定工作负载定制的调度策略可以执行多种优化,如优先执行耗时短、资源占用小的查询.此类特定于工作负载的策略在实践中很少使用,因为它们需要专家知识,并且需要花费大量精力来设计、实现和验证.而借助强化学习的方法,多种多样的查询调度策略可以得以学习和应用.

文献[99-101]提出了适用于云服务环境的查询调度框架 WiSeDB,该框架可以为数据库云服务商提供资源配置、查询分配和查询调度策略,在保障服务质量的同时,减少云服务商的开销.WiSeDB 可以为用户与云服务商提供多种不同的查询性能策略,如最长查询时间限制、平均查询时间限制等.WiSeDB 以一组负载和性能策略为输入,通过决策树模型生成最优的查询调度策略,策略包括查询的执行顺序、是否启用新服务器以及将查询分配到哪台服务器执行.决策树模型的构建是一个离线(Off-line)过程,在决策树构建过程中,通过生成随机的小规模的并发负载,将查询调度问题转换为图中的最短路径问题.在该图中,将某条查询置于某个服务器执行是图上的一个点,边的权重为在该服务器上执行该查询的代价,使用 A*算法寻找最短路径.最后,WiSEDB 通过对最短路径提取其查询调度策略的特征,构建决策树模型,实现了对大规模负载并发的管理.文献[102]提出了 SageDB,在 SageDB 中,同样通过强化学习的方法,以数据分布和查询负载为输入,来对负载调度的神经网络进行训练,以达到优化查询平均完成时间的目标.

Fig.6 Process of workload scheduling using reinforcement learning

图 6 基于强化学习的查询调度的一般过程

4.3 小结

由于强化学习非常适于随机、有延时、有交互的学习场景,因此强化学习的方法对于查询执行过程中的

自适应查询处理和查询调度有天然的适用性,图 6 展示了强化学习和基于强化学习的查询调度的一般过程.通过探索-利用(Exploration-Exploitation)的学习方式,这些方法可以发现和学习到更好的策略,服务于查询执行过程中的自适应查询处理和查询调度优化.

5 查询语言处理

最近,自然语言作为数据库一种查询接口的需求得到了广泛的关注,通过自然语言对数据库进行查询,可以使非专业用户高效地制定复杂的信息请求,降低数据库的使用门槛.同时,利用查询语言嵌入技术(Embedding),可以进行查询语言语法检测与推荐,还可以借此抽取查询负载的特征信息以用于多种优化任务.因此,利用人工智能技术对查询语言进行转换和嵌入得到了一系列研究.

5.1 自然语言与SQL的转换

文献[103]提出了一个关系型数据库工具 DBPal,实现了自然语言到 SQL 查询语句的基于神经网络学习的转换,为用户和数据的交互提供了便捷的操作.DBPal 主要由两种部件组成,一是强大的查询翻译,基于序列到序列的循环神经网络模型的查询转换框架,将不同的自然语言转化成关系型数据库操作;二是交互式的自动补全,给用户提供查询输入建议,自动补全的语言模型,也基于相同的神经网络模型.构建翻译查询的关键是生成训练数据集,DBPal 人工标注 SQL-NL(SQL 和自然语言对)模板,再使用自然语言处理的一些方法进行扩充.DBPal 侧重于自然语言到 SQL 查询语言转换数据集的生成,为了生成均衡的扩充训练数据,DBPal 在给定特定数据库模式的情况下使用贝叶斯优化自动调整生成扩充数据的参数.

文献[104]提出了 Seq2SQL,Seq2SQL 是一个使用强化学习将自然语言转化成 SQL 的工具.Seq2SQL 使用数据库的循环执行来训练一个序列到序列的神经网络的强化学习模型,从而学习从自然语言生成查询的策略.Seq2SQL 的特点是利用了 SQL 语句的固有结构,将自然语言和表的全部列作为输入,将输入使用三个组件分别转化为 SQL 语句的聚合操作,SELECT 操作,和 WHERE 子句,然后填充对应的数据库模式,从而生成完整的 SQL 语句.对于聚合操作和 SELECT 操作,Seq2SQL 使用了交叉熵损失函数,分别生成其自然语言对应的聚合操作和选择的列.WHERE 子句生成使用了基于策略的梯度函数,以解决 WHERE 子句条件的无序性.在训练期间,查询执行的结果被用作训练强化学习算法的奖励.因此,Seq2SQL 使用混合目标进行训练,结合了交叉熵损失和基于策略的梯度函数,此外,Seq2SQL 利用 SQL 的结构来修剪生成的查询空间,显著简化查询语句生成问题.文献[105]提出了 SQLNet,尝试在不使用强化学习的情况下从自然语言生成 SQL.SQLNet 采用基于草图(Sketch)的方法,从草图生成 SQL 查询.草图基于 SQL 的语法结构构建包含依赖图,从而可以通过仅考虑其依赖的先前预测来完成一个预测.SQLNet 使用多个 LSTM 神经网络预测草图中的每一个槽的内容,以填充草图模型生成完整的 SQL 查询.SQLNet 在传统的序列到序列的神经网络模型基础上提出了一个序列到集合的神经网络模型以及列注意力(Column Attention)机制,解决 WHERE 子语句中的无序约束问题和捕获预测时草图中定义的依赖关系.通过这种网络模型,SQLNet 从根本上解决了由于相同的 SQL 查询可能具有多个等效序列,而训练序列到序列的神经网络模型对其中一个的选择很敏感的问题. Seq2SQL 和 SQLNet 都尝试通过解决查询语句中的等效序列的无序问题,从而提升自然语言转化为 SQL 的精度,但是二者略有不同.在生成 SQL 语句方面,Seq2SQL 将 SQL 分为聚合操作等三部分,而 SQLNet 则是基于草图;在网络结构上,Seq2SQL 使用混合目标进行训练序列到序列的网络,结合了交叉熵损失和基于策略的梯度函数,而 SQLNet 结合草图使用了序列到集合的神经网络.从实验结果上看,SQLNet 较 Seq2SQL 在查询匹配精度和查询结果匹配精度均有较大的提升.

5.2 SQL Embedding

文献[106]提出了一种将 SQL 查询表示为向量的方法 Query2Vec,以支持工作负载分析等任务.类似于自然语言处理领域中 Doc2Vec 通过学习表示将文档或句子将向量表示为定长向量的方法,Query2Vec 将此方法用于学习 SQL 查询的向量表示.Query2Vec 通过定性评估发现,相比较使用原始 SQL 语句,使用查询计划和查询模板生成学习向量表示的语义保留能力更强.为了避免设置上下文的大小,文章提出了一种基于 Query2Vec 方

法的长短期记忆(LSTM)的神经网络.网络由编码和解码两部分组成,编码器 LSTM 网络将查询建模为单词序列,并使用编码器 LSTM 最后单元的隐藏状态作为再现输入查询的解码器 LSTM 的输入.一旦网络构建完成,通过将查询传递到编码器网络,在 LSTM 组件上运行正向传递并输出编码器 LSTM 最后单元的隐藏状态作为表示向量.Query2Vec 将上述提出的向量表示 SQL 查询的方法在索引推荐和内存错误预测两个方面进行了应用,并取得了良好的效果.文献[107]提出了 CODE-NN 模型,CODE-NN 训练了带有注意力机制的长短期记忆(LSTM)神经网络模型,可以根据输入的 SQL 语句,输出其可以实现的功能描述.

5.3 小结

将自然语言处理的方法应用到查询语言中,将自然语言自动的转换为数据库查询语言,可以为用户使用数据库提供更为方便和直观的查询接口.将查询语言进行词嵌入,为查询语言的推荐、纠错和负载特征提取等任务提供了新方法.

6 总结与展望

近年来,数据库与人工智能技术的结合得到了广泛的关注,取得了一系列研究成果.本文从数据库负载分析与参数配置、数据存取、查询优化、查询执行、查询语言处理五个方面综述了人工智能赋能的数据管理新技术.人工智能技术与数据库的结合主要有四种方式,第一种是针对传统数据库中带有可微分参数的模型,通过梯度下降等机器学习方法,优化参数以提高模型的性能表现,如对查询规模估算核函数参数的优化[54];第二种是通过特征工程方法,将数据库的状态、负载、性能等进行特征提取,使用机器学习算法进行学习,以预测数据库对新特征的性能表现,如对数据库配置参数和其性能表现进行特征向量化后进行学习,以调整配置参数优化数据库查询性能[19];第三种是使用机器学习模型完全替换传统数据库中的部分组件,如使用神经网络模型替代 B+树和哈希索引等[30];第四种是先使用传统方法在样本上进行操作,再提取传统方法处理过程的特征进行学习,如 WiSeDB[99-101]进行查询调度.最后,本文对人工智能赋能的数据库系统做出以下展望:

• 基于机器学习的查询规模估算虽然在估算的准确率上取得了优于传统方法的表现,但目前的研究仅停留在估算结果的优劣上,而该结果如何影响查询计划的优化过程、对最终查询执行的效率有多大程度的提升仍然有待研究.同时,目前的研究大多没有考虑机器学习模型替换传统方法可能带来的计算时延增加、模型规模变大、模型的可靠性和可解释性等问题.

• 强化学习因为可以解决序列决策问题,而非常适用于数据库中的多种应用场景.在目前使用强化学习思想进行查询优化、查询调度管理等方法中常见的强化学习方法较为老旧,面临收敛速度慢的问题,未来应考虑一些新的强化学习方法如情节性深度强化学习(Episodic DRL)、元强化学习(Meta RL)在数据库优化中的应用.

• 目前很多基于机器学习方法的查询优化技术仍然停留在特征工程和模型选择阶段,没有有效利用数据库中优化任务的特点,如查询执行计划通常以树的结构表示,未来可以探索基于树形结构的 LSTM 和 CNN 以及基于图的 GNN 对查询计划树的学习和表示.

• 复杂的连接操作优化仍然是数据库查询优化中具有挑战性的问题,分别在查询规模估算、查询代价估算、查询计划生成和自适应查询处理等多个阶段对连接操作进行优化会产生怎样的结果,能否将这些过程统一起来仍是值得研究的问题.

• 本文选取了五个方面中有代表性的研究介绍了人工智能赋能的数据库,但人工智能技术在数据库中的应用并不局限于这些研究.数据库故障处理、安全与隐私保护、物化视图选择等诸多方面同样可以应用人工智能技术,实现自动化、高效率、自学习的目标,未来应该继续探索人工智能技术在数据库中全方位的应用.

• 目前对人工智能赋能的数据库系统的研究大多从某个切入点展开,虽然文献[102,108,109]提出了 SageDB, Database Learning, Self-Driving DBMS 等系统化设计方案,但其仍处在雏形阶段.设计和实现兼具自动调优、自主管理的自学习数据库系统应该作为研究的最终目标.

References:

[1] Brown, Kurt P., Michael J. Carey, and Miron Livny. Towards an autopilot in the DBMS performance cockpit. International Workshop on High Performance Transaction Systems, 1993. 26-29.

[2] Weikum G, Hasse C, Mönkeberg A, Zabback P. The COMFORT automatic tuning project. Information systems, 1994,19(5):381-432.

[3] Chaudhuri, Surajit and V. Narasayya. AutoAdmin "what-if" index analysis utility. Acm Sigmod International Conference on Management of Data, 1998,27(2):367-378.

[4] Chaudhuri, Surajit, and Vivek Narasayya. Self-tuning database systems: a decade of progress. Proceedings of the 33rd international conference on Very large data bases, 2007. 3-14.

[5] Stillger M, Lohman GM, Markl V, Kandil M. LEO-DB2's learning optimizer. Proceedings of the international conference on Very large data bases, vol.1, 2001. 19-28.

[6] Zhang N, Haas PJ, Josifovski V, Lohman GM, Zhang C. Statistical learning techniques for costing XML queries. Proceedings of the 31st international conference on Very large data bases, 2005. 289-300.

[7] Popescu AD, Balmin A, Ercegovac V, Ailamaki A. PREDIcT: Towards predicting the runtime of large scale iterative analytics. Proceedings of the international conference on Very large data bases, 2013,6(14):1678-1689.

[8] Zilio DC, Zuzarte C, Lightstone S, Ma W, Lohman GM, Cochrane RJ, Pirahesh H, Colby L, Gryz J, Alton E, Valentin G. Recommending materialized views and indexes with the IBM DB2 design advisor. International Conference on Autonomic Computing, 2004. 180-187.

[9] Kwan E, Lightstone S, Schiefer B, Storm A, Wu L. Automatic database configuration for DB2 Universal Database: Compressing years of performance expertise into seconds of execution. Datenbanksysteme für Business, Technologie und Web, 2003,20: 620-629.

[10] Schnaitter K, Abiteboul S, Milo T, Polyzotis N. On-line index selection for shifting workloads. 2007 IEEE 23rd International Conference on Data Engineering Workshop, 2007. 459-468.

[11] Holze, Marc, and Norbert Ritter. Towards workload shift detection and prediction for autonomic databases. Proceedings of the ACM first Ph. D. workshop in Conference on Information and Knowledge Management, 2007. 109-116.

[12] Holze, Marc, Ali Haschimi, and Norbert Ritter. Towards workload-aware self-management: Predicting significant workload shifts. 2010 IEEE 26th International Conference on Data Engineering Workshops, 2010. 111-116.

[13] Holze, Marc, Claas Gaidies, and Norbert Ritter. Consistent on-line classification of DBS workload events. Proceedings of the 18th ACM conference on Information and knowledge management, 2009. 1641-1644.

[14] Ma L, Van Aken D, Hefny A, Mezerhane G, Pavlo A, Gordon GJ. Query-based workload forecasting for self-driving database management systems. Proceedings of the 2018 International Conference on Management of Data, 2018. 631-645.

[15] Belknap P, Dageville B, Dias K, Yagoub K. Self-tuning for SQL performance in Oracle database 11g. 2009 IEEE 25th International Conference on Data Engineering, 2009. 1694-1700.

[16] Storm AJ, Garcia-Arellano C, Lightstone SS, Diao Y, Surendra M. Adaptive self-tuning memory in DB2. Proceedings of the 32nd international conference on Very large data bases, 2006. 1081-1092.

[17] Tran DN, Huynh PC, Tay YC, Tung AK. A new approach to dynamic self-tuning of database buffers. ACM Transactions on Storage, 2008,4(1): 3.

[18] Duan S, Thummala V, Babu S. Tuning database configuration parameters with iTuned. Proceedings of the VLDB Endowment, 2009, 2(1): 1246-1257.

[19] Van Aken D, Pavlo A, Gordon GJ, Zhang B. Automatic database management system tuning through large-scale machine learning. Proceedings of the 2017 ACM International Conference on Management of Data, 2017. 1009-1024.

[20] Zhang J, Liu Y, Zhou K, Li G, Xiao Z, Cheng B, Xing J, Wang Y, Cheng T, Liu L, Ran M. An end-to-end automatic cloud database tuning system using deep reinforcement learning. Proceedings of the 2019 International Conference on Management of Data, 2019. 415-432.

[21] Agrawal, Sanjay, Vivek Narasayya, and Beverly Yang. Integrating vertical and horizontal partitioning into automated physical database design. Proceedings of the 2004 ACM SIGMOD international conference on Management of data, 2004. 359-370.

[22] Zilio DC, Sevcik KC. Physical database design decision algorithms and concurrent reorganization for parallel database systems. University of Toronto, 1999.

[23] Rao J, Zhang C, Megiddo N, Lohman G. Automating physical database design in a parallel database. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, 2002. 558-569.

[24] Curino C, Jones E, Zhang Y, Madden S. Schism: a workload-driven approach to database replication and partitioning. Proceedings of the international conference on Very large data bases, 2010,3(1-2):48-57.

[25] Karypis G, Kumar V. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 5.0. http://www.cs.umn.edu/~metis, 2009.

[26] C Cowling J, Liskov B. Granola. Granola: Low-overhead distributed transaction coordination. USENIX Annual Technical Conference, 2012. 223-235.

[27] Turcu A, Palmieri R, Ravindran B, Hirve S. Automated data partitioning for highly scalable and strongly consistent transactions. IEEE transactions on parallel and distributed systems, 2015,27(1):106-118.

[28] Durand GC, Pinnecke M, Piriyev R, Mohsen M, Broneske D, Saake G, Sekeran MS, Rodriguez F, Balami L. GridFormation: Towards self-driven online data partitioning using reinforcement learning. Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management，2018.

[29] Hilprecht B, Binnig C, Roehm U. Learning a partitioning advisor with deep reinforcement learning. arXiv preprint arXiv:1904.01279 (2019).

[30] Kraska T, Beutel A, Chi EH, Dean J, Polyzotis N. The case for learned index structures. Proceedings of the 2018 International Conference on Management of Data, 2018. 489-504.

[31] Schkolnick, M. The optimal selection of secondary indices for files. Information Systems, 1975, 1(4):141-146.

[32] Stonebraker, M. The choice of partial inversions and combined indices. International Journal of Computer & Information Sciences, 1974, 3(2):167-188.

[33] Finkelstein S, Schkolnick M, Tiberio P. Physical database design for relational databases. ACM Transactions on Database Systems, 1988, 13(1):91-128.

[34] Whang, K. Y. Separability—An approach to physical database design. IEEE transactions on computers, 1984, 100(3):209-222.

[35] Chaudhuri S, Narasayya VR. An efficient, Cost-driven index selection tool for Microsoft SQL server. Proceedings of the international conference on Very large data bases, 1997. 146-155.

[36] Hammer M, Chan A. Index selection in a self-adaptive data base management system. Proceedings of the 1976 ACM SIGMOD international conference on Management of data, 1976. 1-8.

[37] Sattler KU, Geist I, Schallehn E. Quiet: Continuous query-driven index tuning. Proceedings of the 29th international conference on Very large data bases-Volume 29, 2003. 1129-1132.

[38] Schnaitter K, Abiteboul S, Milo T, Polyzotis N. On-line Database Tuning. Technical Report UCSC-CRL-06-07, UC Santa Cruz, 2006.

[39] Idreos S, Kersten ML, Manegold S. Updating a cracked database. Proceedings of the 2007 ACM SIGMOD international conference on Management of data, 2007. 413-424.

[40] Aouiche K, Darmont J. Data mining-based materialized view and index selection in data warehouses. Journal of Intelligent Information Systems, 2009, 33(1): 65-93.

[41] Sharma A, Schuhknecht F M, Dittrich J. The case for automatic database administration using deep reinforcement learning. arXiv preprint arXiv:1801.05643 (2018)

[42] Ding B, Das S, Marcus R, Wu W, Chaudhuri S, Narasayya VR. AI meets AI: Leveraging query executions to improve index recommendations. Proceedings of the 2019 International Conference on Management of Data, 2019. 1241-1258.

[43] Mannino MV, Chu P, Sager T. Statistical profile estimation in database systems. ACM Computing Surveys,1988,20(3):191-221.

[44] Kooi RP. The optimization of queries in relational databases. 1981. 4184-4184.

[45] Ioannidis Y . The history of histograms (abridged). Proceedings of the international conference on Very large data bases. Morgan Kaufmann, 2003. 19-30.

[46] Selinger PG, Astrahan MM, Chamberlin DD, Lorie RA, Price TG. Access path selection in a relational database management system. Proceedings of the 1979 ACM SIGMOD international conference on Management of data, 1979. 23-34.

[47] Fedorowicz J. Database evaluation using multiple regression techniques. ACM SIGMOD Record. 1984,14(2):70-76.

[48] Lefons E, Silvestri A, Tangorra F. An analytic approach to statistical databases. Proceedings of the international conference on Very large data bases, 1983. 260-274.

[49] Sun W, Ling Y, Rishe N, Deng Y. An instant and accurate size estimation method for joins and selections in a retrieval-intensive environment. ACM SIGMOD Record. 1993,22(2):79-88.

[50] Lipton RJ, Naughton JF, Schneider DA. Practical selectivity estimation through adaptive sampling. ACM SIGMOD Record. 1990,19(2):1-11.

[51] Piatetsky-Shapiro G, Connell C. Accurate estimation of the number of tuples satisfying a condition. ACM SIGMOD Record, 1984,14(2):256-276.

[52] Chen CM, Roussopoulos N. Adaptive selectivity estimation using query feedback. ACM SIGMOD Record, 1994,23(2):161-172.

[53] Gunopulos D, Kollios G, Tsotras VJ, Domeniconi C. Selectivity estimators for multidimensional range queries over real attributes. The international journal on very large data base, 2005,14(2):137-154.

[54] Heimel M, Kiefer M, Markl V. Self-tuning, Gpu-accelerated kernel density models for multidimensional selectivity estimation. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015. 1477-1492.

[55] Kiefer M, Heimel M, Breß S, Markl V. Estimating join selectivities using bandwidth-optimized kernel density models. Proceedings of the international conference on Very large data bases, 2017,10(13):2085-2096.

[56] Malik T, Burns RC, Chawla NV. A black-box approach to query cardinality estimation. The Conference on Innovative Data Systems Research, 2007. 56-67.

[57] Wu C, Jindal A, Amizadeh S, Patel H, Le W, Qiao S, Rao S. Towards a learning optimizer for shared clouds. Proceedings of the international conference on Very large data bases, 2018,12(3):210-222.

[58] Lakshmi S, Zhou S. Selectivity estimation in extensible databases-a neural network approach. Proceedings of the international conference on Very large data bases, 1998. 623-627.

[59] Liu H, Xu M, Yu Z, Corvinelli V, Zuzarte C. Cardinality estimation using neural networks. Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering, 2015. 53-59.

[60] Selmic RR, Lewis FL. Neural-network approximation of piecewise continuous functions: Application to friction compensation. IEEE transactions on neural networks, 2002,13(3):745-751.

[61] Dutt A, Wang C, Nazi A, Kandula S, Narasayya V, Chaudhuri S. Selectivity estimation for range predicates using lightweight models. Proceedings of the international conference on Very large data bases, 2019,12(9):1044-1057.

[62] Kipf A, Kipf T, Radke B, Leis V, Boncz P, Kemper A. Learned cardinalities: estimating correlated joins with deep learning. ArXiv abs/1809.00677 (2018)

[63] Kipf A, Vorona D, Müller J, Kipf T, Radke B, Leis V, Boncz P, Neumann T, Kemper A. Estimating cardinalities with deep sketches. Acm Sigmod International Conference on Management of Data, 2019. 1937-1940.

[64] Marcus R, Papaemmanouil O. Flexible operator embeddings via deep learning. ArXiv abs/1901.09090 (2019)

[65] 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. 2009 IEEE 25th International Conference on Data Engineering, 2009. 592-603.

[66] Akdere M, Cetintemel U, Riondato M, Upfal E, Zdonik SB. The case for predictive database systems: opportunities and challenges. The Conference on Innovative Data Systems Research, 2011. 167-174.

[67] Akdere M, Çetintemel U, Riondato M, Upfal E, Zdonik SB. Learning-based query performance modeling and prediction. 2012 IEEE 28th International Conference on Data Engineering, 2012. 390-401.

[68] Li J, König AC, Narasayya V, Chaudhuri S. Robust estimation of resource consumption for sql queries using statistical techniques. Proceedings of the international conference on Very large data bases, 2012,5(11):1555-1566.

[69] Marcus R, Papaemmanouil O. Plan-Structured deep neural network models for query performance prediction. Proceedings of the international conference on Very large data bases, 2019,12(11):1733-1746.

[70] Gupta C, Mehta A, Dayal U. PQR: Predicting query execution times for autonomous workload management. 2008 International Conference on Autonomic Computing, 2008. 13-22.

[71] Duggan J, Cetintemel U, Papaemmanouil O, Upfal E. Performance prediction for concurrent database workloads. Acm Sigmod International Conference on Management of Data, 2011. 337-348.

[72] Ahmad M, Duan S, Aboulnaga A, Babu S. Predicting completion times of batch query workloads using interaction-aware models and simulation. Proceedings of the 14th International Conference on Extending Database Technology, 2011. 449-460.

[73] Ahmad M, Aboulnaga A, Babu S, Munagala K. Interaction-aware scheduling of report-generation workloads. The international journal on very large data base, 2011,20(4):589-615.

[74] Wu W, Chi Y, Hacígümüş H, Naughton JF. Towards predicting query execution time for concurrent and dynamic database workloads. Proceedings of the international conference on Very large data bases, 2013,6(11):925-936.

[75] Duggan J, Papaemmanouil O, Cetintemel U, Upfal E. Contender: A resource modeling approach for concurrent query performance prediction. The International Conference on Extending Database Technology. 2014. 109-120.

[76] Färber F, Cha SK, Primsch J, Bornhövd C, Sigg S, Lehner W. SAP HANA database: Data management for modern business applications. ACM Sigmod Record, 2012,40(4):45-51.

[77] Ibaraki, Toshihide, and Tiko Kameda. On the optimal nesting order for computing n-relational joins. ACM Transactions on Database Systems, 1984,9(3): 482-502.

[78] Selinger PG, Astrahan MM, Chamberlin DD, Lorie RA, Price TG. Access path selection in a relational database management system. Proceedings of the 1979 ACM SIGMOD international conference on Management of data. 1979. 23-34.

[79] Meister, Andreas, Guido Moerkotte, and Gunter Saake. Errata for Analysis of two existing and one new dynamic programming algorithm for the generation of optimal bushy join trees without cross products. Proceedings of the international conference on Very large data bases, 2018,11(10):1069-1070.

[80] Swami, Arun. Optimization of large join queries: Combining heuristics and combinatorial techniques. ACM SIGMOD Record. ACM, 1989,18(2):367-376.

[81] Fegaras, Leonidas. A new heuristic for optimizing large queries. International Conference on Database and Expert Systems Applications. 1998. 726-735.

[82] Leis V, Gubichev A, Mirchev A, Boncz P, Kemper A, Neumann T. How good are query optimizers, really?. Proceedings of the international conference on Very large data bases, 2015,9(3):204-215.

[83] Dieu N, Dragusanu A, Fabret F, Llirbat F, Simon E. 1, 000 tables inside the from. Proceedings of the international conference on Very large data bases, 2009,2(2): 1450-1461.

[84] Krishnan S, Yang Z, Goldberg K, Hellerstein J, Stoica I. Learning to optimize join queries with deep reinforcement learning. arXiv preprint arXiv:1808.03196 (2018).

[85] Marcus, Ryan, and Olga Papaemmanouil. Deep reinforcement learning for join order enumeration. Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, 2018. 3.

[86] Ortiz J, Balazinska M, Gehrke J, Keerthi SS. Learning state representations for query optimization with deep reinforcement learning. arXiv preprint arXiv:1803.08604 (2018).

[87] Marcus, Ryan and Olga Papaemmanouil. Towards a hands-free query optimizer through deep learning. ArXiv abs/1809.10212 (2018).

[88] Marcus R, Negi P, Mao H, Zhang C, Alizadeh M, Kraska T. Neo: A learned query optimizer. arXiv preprint arXiv:1904.03711 (2019).

[89] Mou L, Li G, Zhang L, Wang T, Jin Z. Convolutional neural networks over tree structures for programming language processing. Thirtieth AAAI Conference on Artificial Intelligence, 2016. 1287-1293.

[90] Babu, Shivnath, and Pedro Bizarro. Adaptive query processing in the looking glass. Proceedings of the Second Biennial Conference on Innovative Data Systems Research Jan. 2005.

[91] Deshpande, Amol, Zachary Ives, and Vijayshankar Raman. "Adaptive query processing." Foundations and Trends® in Databases, 2007,1(1):1-140.

[92] Rundensteiner EA, Ding L, Sutherland T, Zhu Y, Pielech B, Mehta N. Cape: Continuous query engine with heterogeneous-grained adaptivity. Proceedings of the Thirtieth international conference on Very large data bases, 2004,30:1353-1356.

[93] Avnur, Ron, and Joseph M. Hellerstein. Eddies: Continuously adaptive query processing. ACM SIGMOD Record. ACM, 2000,29(2):261-272.

[94] Waldspurger, Carl A., and William E. Weihl. Lottery scheduling: Flexible proportional-share resource management. Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation, 1994. 1.

[95] Trummer I, Wang J, Maram D, Moseley S, Jo S, Antonakakis J. SkinnerDB: Regret-bounded query evaluation via reinforcement learning. Proceedings of the international conference on Very large data bases, 2018,11(12):2074-2077.

[96] Răducanu, Bogdan, Peter Boncz, and Marcin Zukowski. Micro adaptivity in vectorwise. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, 2013. 1231-1242.

[97] Zukowski, Marcin and Peter A. Boncz. Vectorwise: Beyond Column Stores. IEEE Data Engineering Bulletin 35, 2012. 21-27.

[98] Kaftan T, Balazinska M, Cheung A, Gehrke J. Cuttlefish: A lightweight primitive for adaptive query processing. arXiv preprint arXiv:1802.09180 (2018).

[99] Marcus, Ryan, and Olga Papaemmanouil. Workload management for cloud databases via machine learning. 2016 IEEE 32nd International Conference on Data Engineering Workshops, 2016. 27-30.

[100] Marcus, Ryan, and Olga Papaemmanouil. WiSeDB: A learning-based workload management advisor for cloud databases. Proceedings of the international conference on Very large data bases, 2016,9(10): 780-791.

[101] Marcus, Ryan, Sofiya Semenova, and Olga Papaemmanouil. A learning-based service for cost and performance management of cloud databases. 2017 IEEE 33rd International Conference on Data Engineering, 2017. 1361-1362.

[102] Kraska T, Alizadeh M, Beutel A, Chi H, Ding J, Kristo A, Leclerc G, Madden S, Mao H, Nathan V. SageDB: A learned database system. The Conference on Innovative Data Systems Research, 2019.

[103] Utama P, Weir N, Basik F, Binnig C, Cetintemel U, Hättasch B, Ilkhechi A, Ramaswamy S, Usta A. An End-to-end neural natural language interface for databases. arXiv preprint arXiv:1804.00401 (2018).

[104] Zhong V, Xiong C, Socher R. Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv:1709.00103 (2017).

[105] Xu, Xiaojun, Chang Liu, and Dawn Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. arXiv preprint arXiv:1711.04436 (2017).

[106] Jain S, Howe B, Yan J, Cruanes T. Query2Vec: An evaluation of NLP techniques for generalized workload analytics. arXiv preprint arXiv:1801.05613 (2018).

[107] Iyer S, Konstas I, Cheung A, Zettlemoyer L. Summarizing source code using a neural attention model. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). (2016): 2073-2083.

[108] Park Y, Tajik AS, Cafarella M, Mozafari B. Database learning: Toward a database that becomes smarter every time. Proceedings of the 2017 ACM International Conference on Management of Data, 2017. 587-602.

[109] Pavlo A, Angulo G, Arulraj J, Lin H, Lin J, Ma L, Menon P, Mowry TC, Perron M, Quah I, Santurkar S. Self-Driving database management systems. The Conference on Innovative Data Systems Research, 2017.