基于机器学习的数据库技术综述

李国良 周煊赫 孙 佶 余 翔 袁海涛 刘佳斌 韩 越

(清华大学计算机系 北京 100084)

摘 要 大数据时代下,面对不断膨胀的数据信息、复杂多样的应用场景、异构的硬件架构和参差不齐的用户使用水平,传统数据库技术很难适应这些新的场景和变化.机器学习技术因其较强的学习能力,逐渐在数据库领域展现出了潜力和应用前景.论文首先给出一个高效、高可靠、高可用、自适应性强的数据库系统需要涵盖的方面,包括数据库运维、数据存储、查询优化等.其次,讨论机器学习算法与数据库技术结合过程中可能面临的挑战,包括训练数据少、训练时间长、泛化能力有限、适应性差四个方面.然后,综述数据库技术与机器学习结合的现状以及具体技术.其中,重点介绍数据库自动调参、查询基数估计、查询计划选择、索引和视图自动选择五个方向.自动调参技术包括启发式算法、传统机器学习、深度强化学习三类.启发式算法从离散的参数空间中通过抽样探索最优子空间,可以有效提高调参效率,但是难以保证在有效资源限制内找到合适配置;传统机器学习算法在经过降维的参数空间中学习系统状态到指定负载模板的映射关系,一定程度上提升模型的适应性;深度强化学习在高维参数空间中迭代的学习调优策略,并利用神经网络提升对高维数据的处理能力,有效降低训练数据的需求.查询基数估计包括面向查询和面向执行计划两类.面向查询方法利用卷积神经网络学习表数据、查询条件、连接条件之间的关系,然而在不同场景下需要大量训练而且泛化能力差;面向执行计划方法在物理算子层面做级联的代价估计,一定程度上提高对不同查询的适应能力.查询计划选择包括深度学习和强化学习两类.深度学习方法融合数据库估计器的代价值和数据特征,提高对每种计划代价估计的精度,但是结果严重依赖估计器的表现;强化学习基于最终目标迭代生成查询计划,降低方法对查询代价的依赖性.自动索引推荐包括分类器、强化学习、遗传算法三类.分类算法根据离散的表特征分析不同索引的创建开销和效率,通过结合遗传算法,提高对复合索引的推荐效率;强化学习进一步提供增量式索引推荐的效率,实现在线索引选择.自动视图选择包括启发式算法、概率统计、强化学习三类.启发式算法通过在视图构建的有向无环图上做贪心探索,提高选择效率,然而适应性差;基于概率统计的算法将视图选择形式化成一个 0-1 选择问题,有效降低图的探索开销;强化学习方法将视图的创建和删除统一成动态选择过程,基于强化学习的训练策略进一步提高选择效率.最后,从八个方面展望机器学习将给数据库带来的革命性突破.

关键词 数据库;机器学习;强化学习;深度学习;查询优化

中图法分类号 TP392 **DOI 号** 10.11897/SP.J.1016.2020.02019

A Survey of Machine Learning Based Database Techniques

LI Guo-Liang ZHOU Xuan-He SUN Ji YU Xiang YUAN Hai-Tao LIU Jia-Bin HAN Yue

(*Department of Computer Science, Tsinghua University, Beijing* 100084)

Abstract In the era of big data, for the ever-expanding data volume, complex and diverse application scenarios, heterogeneous hardware architecture and different types of users, traditional database techniques cannot adapt to these new scenarios and changes. So machine learning, known for its learning ability, gradually shows potential and application prospects in database.

收稿日期:2019-06-01;在线发布日期:2019-11-05.本课题得到国家自然科学基金(61632016)、国家"九七三"重点基础研究发展规划项目基金(2015CB358700)资助.**李国良**,博士,教授,中国计算机学会(CCF)会员,主要研究领域为数据库、大数据等.E-mail:liguoliang@tsinghua.edu.cn.**周煊赫**,博士研究生,主要研究方向为数据库与人工智能的交叉技术.**孙 佶**,博士研究生,主要研究方向为字符串相似度匹配、数据库与人工智能的交叉技术.**余 翔**,博士研究生,主要研究方向为数据库与人工智能的交叉技术.**袁海涛**,博士研究生,主要研究方向为路网、数据库与人工智能的交叉技术.**刘佳斌**,硕士研究生,主要研究方向为数据库与人工智能的交叉技术.**韩 越**,博士研究生,主要研究方向为数据库与人工智能的交叉技术.

Based on full investigation and analysis, we first summarize the requirements of machine learning for building an efficient, reliable, highly available and adaptive database system, including database operation and maintenance, data storage, optimizer and executor, query optimization, database workload management, database security and privacy, database self-management, database for machine learning. Then, we discuss the potential challenges in the process of combining machine learning algorithms with database techniques from four aspects, including lack of training data, long training time, limited generalization ability, and challenges in applying machine learning models with specific database problems. Next, we survey the researches of machine-learning-based techniques, including automatic parameter tuning, automatic cardinality estimation, automatic query plan selection, automatic index and view selection. Automatic tuning technology includes heuristic algorithm, traditional machine learning and deep reinforcement learning. Heuristic algorithms explore the optimal subspace through sampling from the discrete parameter space, which can effectively improve the efficiency of parameter tuning, but they are difficult to find the appropriate configuration within the resource limit; traditional machine learning algorithm learns the mapping relationship between the system state and the specified workload template in the reduced dimension parameter space, which improves the adaptability of the model; deep reinforcement learning iteratively learns the optimization strategy in the high-dimensional parameter space, and uses neural network to improve the processing ability of high-dimensional data. It can effectively reduce the demand of training data; automatic cardinality estimation includes query-oriented method and query-plan-oriented method. The former uses convolutional neural network (CNN) to learn the relationship among data, filter conditions and join conditions. However, it is poor in generalization for different datasets. The latter estimates cardinality of physical operators in cascades, which improves the adaptability to different queries. Query plan selection includes deep learning and reinforcement learning. The deep learning method integrates the estimated cost values and data characteristics, which improve the accuracy of each plan cost estimation, but the results depend heavily on the accuracy of the estimator; deep reinforcement learning method iteratively generates the query plan based on the final goal, and it reduces the dependence on query cost. Automatic index selection includes classifier, reinforcement learning and genetic algorithm: the classification algorithm analyzes the cost of building indexes and the efficiency of different indexes based on the table characteristics. By combining the genetic algorithm, it improves the recommendation efficiency of composite index; reinforcement learning realizes online index selection by incrementally recommending indexes. Automatic view selection includes heuristic algorithm, probability statistics and reinforcement learning. Heuristic algorithms improve selection efficiency by greedily exploring directed acyclic graph of candidate views, but its adaptability is poor. Statistics-based methods formalize view selection into a 0-1 selection problem, effectively reducing the exploration cost of graph. Reinforcement learning methods model the creation and deletion of view into a dynamic selection process, and further improve selection efficiency with a try-and-error training pattern. Finally, we provide the revolutionary breakthroughs that machine learning technologies will bring to databases from eight perspectives.

Keywords database; machine learning; deep learning; reinforcement learning; query optimization

1　引　言

1.1　研究背景

传统数据库技术往往依赖于启发式算法或者人工干预，例如数据库参数调优、故障诊断、索引推荐等.然而在大数据时代，数据库实例越来越多、场景越来越复杂、数据量越来越大，导致传统数据库技术难以满足大数据的需求.例如云数据库具有百万级别的数据库实例，各个实例的应用场景、用户的使用水平都可能有很大差别，直接使用传统启发式算法难以取得令人满意的结果，而人工干预也很难管理这么多的实例.

机器学习通过从历史数据和行为中分析、学习找到更好的设计方案，可以解放很多繁杂的手工工作.因此机器学习技术已经被广泛的运用到多个科研和生产领域[1-3].机器学习也为数据库优化技术带来了新的机遇.传统机器学习模型，如线性感知器、随机森林、支持向量机和集成学习等，能够在历史数据中积累“经验”，提高模型解决复杂问题的能力.但是由于传统机器学习模型相对简单(如线性感知器、回归模型等)，往往只能从单一的层次描述学习过程，严重限制了其应用场景和优化能力.比如，简单的线性回归模型很难解决像数据库调参这样有着高维参数、连续空间的问题[4].

随着算法的改进、计算能力的提升、大数据的变革，深度学习和强化学习为数据库优化技术带来了更大的优化空间.深度学习的核心思想是利用神经网络强大的学习能力，通过调整神经元之间的连接来拟合复杂的高维映射关系[5].这类方法往往适用于传统方法难以解决(如 NP-hard 问题)并存在大量训练数据的场景.数据库有着复杂组件，并有大量用户提供访问数据，为深度学习与数据库优化技术的结合提供了很好的先决条件.强化学习通过与环境进行交互获得的奖赏指导行为，以“试错”的方式进行学习，不断修改从状态到动作的映射策略，以达到优化系统性能的目的.通过结合深度学习和强化学习，深度强化学习模型能够更好地适应工业界复杂的应用场景[6-8].例如在调参场景下，Li 等人[8]提出了一种基于深度强化学习的数据库调优系统，它能利用少量样本高效地迭代学习，在高质量样本匮乏的场景下仍然保持较好的性能.

1.2　研究问题

如图 1 所示，本文从构建一个高效、高可靠、高可用、自适应性强的数据库系统角度出发，从下面八个方面来总结基于机器学习的数据库技术：

图 1　基于机器学习的数据库技术

1.2.1　数据库运维

数据库运维指为保证数据库稳定工作而开展的性能监控、配置优化、故障处理等服务.

(1)性能监控.监控数据库系统的状态，如每秒批量请求数、用户连接数、网络收发效率等.类似于监控人的健康指标，性能监控需要尽可能全面地反映数据库的基本性能，大体上包括实时监控、定时监控、分析报告、跟踪和收集数据四类方法.性能监控不仅需要实时统计指标，而且需要根据这些指标理解系统的运作状况.

(2)配置优化.数据库配置中涉及数百个可调的系统参数，控制着数据库组件的多方面表现.配置优化技术[4,8-10]通过选择合适的参数组合，提高数据库对当前场景的适应性.比如通过为各类缓存区分配合适的内存空间，尽可能减少磁盘 IO 次数；通过更新优化器估计指标提高计划选择的表现等.

(3)故障处理与恢复.故障处理包括用于解决事务、系统和硬件介质故障的各种策略.如当出现数据错误、运算溢出或违反某些完整性限制的情况时，系统需要及时撤销相应事务.故障恢复则指在发生某种故障使数据库状态异常时，把数据恢复到上一正确的历史状态(镜像).目前数据库最常用的恢复方法是预先记录操作日志，在发生故障时根据日志回滚.

(4)容灾备份.在系统出现故障时仍然保证系统能正常运行.一套数据往往需要在多个物理节点

上备份,一旦一处节点因为意外停止工作,整个负载可以迁移到另一处,仍然保证服务.

此外,数据库运维还包括数据迁移、版本管理、统计信息管理等诸多方面.分布式场景下,多硬件集群和高负载压力会导致故障率激增,给数据库运维带来很大的挑战.

1.2.2 数据存储

数据存储涉及到数据的组织、存储和管理方式.当前新的存储方式层出不穷.仅从软件和架构两个角度总结,这方面的研究主要包括存储模型和数据扩容两个方面.

(1)存储模型.对于关系型数据,存储模型分成行式存储和列式存储;对于非关系型数据,存储模型还包括键值对、图和时间序列等.存储模型的选择主要与负载和数据类型相关.目前负载类型主要包括联机事务处理(OLTP)、联机分析处理(OLAP)和混合事务/分析处理(HTAP)三类.如对于 OLAP 型负载,选择列式存储可以提高聚合操作的执行效率.而不同的数据类型对存储模型的要求也可能不同,例如采用表结构存储流数据会导致大量的空间浪费.但是目前已知数据库中还没有实现存储模型的自动选择,我们需要解决根据历史负载和当前数据自动选择存储模型的问题.

(2)数据扩容.弹性扩容是对已有集群进行动态的在线扩容.通过增减节点,弹性扩容可以有效解决预分配空间不足的问题.但是传统行式数据库往往很难支持在线动态扩容.首先,行式存储不仅模式固定,而且维护着大量的索引和视图信息,传统增加列的方法需要用新表替换旧表,开销很大.其次,行式存储下需要将同一张表拆分成多个数据块,分发到多个节点上,但是这些数据块有很大的可能性会被一起访问,会严重加重节点间通信的负担.尽管列式存储水平扩容非常方便,但是集群环境下事务处理效率较低.而且两种方法都有负载均衡的问题,需要研究动态扩容方法.

1.2.3 优化器与执行器

优化器主要负责为查询语句生成相应的执行计划,执行器则根据选择的计划实际进行各类物理操作,因而优化器和执行器表现对于数据库性能好坏尤为重要.这方面的研究工作主要包括代价基数估计、计划选择和分布式协同计算三个方面.

(1)基数估计.不论是用哪种算法来选择执行计划,都必须依赖于代价估计技术,其重要性有时甚至大于执行计划选择算法本身.传统的数据库代价估计主要由两部分构成,基数估计器(cardinality estimator)和代价模型(cost model).目前最先进的用于生产环境中的基数估计器依然存在很大误差.误差甚至能够达到几个数量级,严重影响计划选择的表现.至于代价模型,尽管它一般以估计出来的基数作为操作代价最主要的参考依据,但是还需要根据估计基数和如硬件开销、缓存情况等其他的因素给出操作代价,也是一件复杂的事情.

(2)计划选择.对于一个 SQL 查询语句,数据库的计划选择器会生成相应的连接计划.不同的查询计划对于查询效率有非常大的影响.而且它的状态空间的大小关于表的连接数量是指数级的,找到最优查询计划是一个 NP-hard 问题.静态计划选择方法以数据库的估计器为衡量指标,先用不同的算法(启发式算法、动态规划等)在状态空间中抽样,最后选择估计代价最低的计划.Tzoumas 等人[11]提出了基于强化学习的计划选择方法,通过试错和迭代反馈机制来优选查询计划.

(3)分布式协同机制.在分布式集群上,数据划分和任务调度尤为重要.数据划分不均匀会造成负载倾斜,导致系统性能下降.因此我们需要研究智能数据划分,根据硬件状态来自动进行数据划分.此外,因为不同工作节点经常要共同分担来自同一个 SQL 的执行任务,有效的协同机制需要同时考虑网络开销和计算开销,均衡两个方面的表现.

1.2.4 查询优化

查询优化主要是关于 SQL 层面上的优化工作,旨在写出执行时友好的 SQL 语句.这方面工作主要有 SQL 重写、索引推荐和自然语言查询三个方向.

(1)SQL 重写.SQL 重写是通过改变 SQL 的书写方式,帮助优化器选择更加高效的执行计划.从 SQL 的整个处理流程来看,可以分成外部 SQL 重写和内部 SQL 重写.外部 SQL 重写发生在把 SQL 传递给数据库执行前.这个阶段往往由人手动完成.他们根据一定的 SQL 书写原则对 SQL 进行重写,如尽量避免全表扫描(Full Table Scan);选有索引的列作连接等.依赖人手工完成的 SQL 优化有很大的局限性.首先,书写高效的 SQL 不仅需要一般原则,还需要对相关数据和数据库有充分的理解,对人来说费时费力.其次,一条 SQL 往往有很多种优化方案,而人的认知往往会受到当前 SQL 结构的影响,不能找到最优的 SQL 写法.内部 SQL 重写则在优化器内完成.重写器会对解析出来的 SQL 树进行修改和替换,如用已有的物化视图代替原始表;去

掉多余的连接操作.传统重写器只会做节点级别的修改,即去除无用操作,替换已有的临时表.但是如果重写器可以再自动对 SQL 结构进行分析和优化,应该会有更好的效果.

(2)索引推荐.索引对于提升复杂数据集上检索任务的效率有着非常重要的意义.尤其是在处理事务型负载时,建在合适列上的索引可以大大提升处理效率[12].因此我们需要有效的索引技术来支持数据库操作.但是传统数据库中常用的索引都是通用的数据结构,它们没有对数据的分布进行分析和利用,也没有利用现实世界中更流行更普遍的模型,如深度学习模型.此外,数据集的增长也会导致索引大小的增长,存在空间浪费的问题.

(3)视图推荐.给定一个查询集合,抽取其中的高频子查询,并建立物化视图来提升查询效率,对于提高数据库表现有着很大的帮助.一般而言,选择子查询建立物化视图主要分为两大步骤:识别等价子查询和选择子查询建立物化视图.我们首先识别出不同查询语句之间的等价子查询,作为建立子查询物化视图的候选集.然后我们根据实际情况中出现的约束条件构建子查询的选择优化问题,设计算法建立物化视图,从而提高批量执行 SQL 查询的效率.

1.2.5　数据库负载管理

数据库负载管理对于数据库的执行效率和系统资源的合理配置有很重要的意义,它主要包括负载分析、负载调度和负载预测与生成三个方面.

(1)负载分析.负载分析技术是各类负载调度机制的基础.数据库中往往会从多个级别监控负载的运行情况和资源开销,并记录在系统视图中.一方面,这些信息帮助系统尽可能充分利用系统资源,避免出现内存溢出、CPU 超负荷等资源不足的问题.另一方面,预估查询代价可以帮助系统更合理地进行资源调度.如基于读/写记录代价等物理信息.传统的代价模型多基于经验公式,对不同的物理环境适应能力较差.因而可以利用机器学习模型对当前负载情况和未来负载开销做分析和评估,基于梯度变化对外界环境动态适应.

(2)负载调度.负载调度技术涉及数据库系统的多个方面,主要包括租户管理、资源调度、排队控制和类型隔离.租户管理以用户为单位进行系统资源的分配和隔离.针对传统优先级控制方法中的问题,未来租户管理技术需要在保证高优先级用户服务质量的同时均衡任务对资源的使用.资源调度主要指从用户、任务、查询等多个级别进行资源控制.传统数据库资源调度严重依赖于资源控制相关的参数,如工作区大小、最大并发数目等.但是这些参数是静态的,不合理的配置会导致查询执行失败,如缓冲区溢出等,严重影响数据库性能.因此需要利用强化学习等方法提供合理、健壮的调度机制.排队控制主要发生在某些系统资源的利用率达到指定阈值.通过限制允许同时执行的查询数目,我们一方面要保证高优先级事务的执行,另一方面要避免超过系统负荷,导致内存溢出、死锁等故障.类型隔离主要指对于 OLTP、OLAP 等不同的负载类型,需要采用不同的存储和执行方式.传统数据库多只专门负责一类负载类型,如 MySQL 主要负责事务处理、MongoDB 主要负责事务分析等.但是很多场景下,OLAP 需要 OLTP 产生的数据作为源,这时在不同数据库之间进行数据传输开销过大,需要混合事务处理和分析(HTAP)的数据库,统一调度 OLTP 和 OLAP 任务.

(3)负载预测和生成.负载预测技术基于当前负载的变化规律对未来的变化情况进行分析和预测,对于负载调度、资源预分配和系统配置都有着很重要的意义.但在实际环境下,负载的变化情况是多样的.传统负载预测方法往往由数据库专家根据统计数据对负载特征进行总体把握,不能保证很高的准确性.Ma 等人[13]提出用机器学习的方法进行负载预测.因为在不同硬件环境上使用 CPU、IO 等系统资源作为负载量度差异很大,所以他们提出用逻辑量度对负载特征量化,从而保证预测的稳定性.而负载生成技术的主要目的是生成高质量的负载样本,对测试和机器学习训练有很重要的意义.传统负载样本主要从更大的查询集合中随机取样组合.这种方式导致生成的样本很难具有现实负载的查询分布特性,随机噪声较多.而利用机器学习技术,如对抗生成网络(GAN)的 Generator-Discriminator 机制,可以由真实负载快速生成大量含典型分布特点的样本.此外,利用利普希茨(Lipschitz continuity)等正则化方法,可以提高负载样本的多样性.

1.2.6　数据库安全与隐私

数据库安全的基本目标是利用信息安全和密码学技术,实现数据库数据的保密性、完整性和可用性保护,拒绝非授权的访问,保证数据库系统的运行安全.

(1)智能数据隐藏.根据数据的特点,智能隐藏隐私数据,如身份证号等.数据隐藏技术主要包括插入和替换两种方法.插入方法通过在要隐藏数

据前加一段冗余数据实现数据隐藏，接收方需要根据特殊标识的位置信息找到隐藏的数据.而替换方法通过字节替换或改变字节顺序实现数据隐藏，接收方需要根据事先协商好的规则获得隐藏的数据.现在有很多数据发现工具，能够根据数据异常解析出隐藏的数据.归结原因还是因为隐藏算法多使用简单的数据变化.智能隐藏技术则利用神经网络内部高维的非线性数据变换机制帮助提高数据隐藏的效果，比如在斯坦福大学与谷歌的研究项目中，就曾将卫星图片转换成不易被察觉的高频信号.

(2)智能审计.从数据预处理和动态分析两个方面优化审计工作.首先，传统审计工作往往需要审计人员获取大量业务数据，在对这些数据特征有了较深理解之后，才能真正开展审计工作.智能审计将提取信息的工作交给机器学习模型，不仅能节约人力开销，而且可以帮助审计人员更好地决策.其次，智能审计还要求机器学习模型能够根据用户行为来智能防范用户对数据的窃取，基于时间序列预先防范越权访问、修改等非法行为.

(3)安全漏洞自检测.能够自动发现系统的漏洞.已知安全漏洞检测主要通过安全扫描检索，而未知安全漏洞还需要做大量的测试工作，已有的未知安全漏洞检测技术包括源代码扫描、反汇编扫描、环境错误注入分析等等.其中软件环境错误注入分析虽然是一种动态的检测技术，仍依赖于系统已知的安全缺陷.Ritter 等人[14]提出利用机器学习训练算法从百万条推文中发现安全漏洞，不仅能检测出国家漏洞数据库中大多数的已知漏洞，而且能够对潜在漏洞进行预测和评级.

1.2.7 数据库自管理

数据库自管理技术主要指自动检测和修复数据库中的各类软件、硬件和架构问题.其基本目标是从管理层面提高数据库的稳定性.当前主要有自诊断、自恢复和可视化管理工具三个方面.

(1)自诊断.自诊断指能够自动诊断数据库的状态.一方面，要求实时监控可能发生的数据库损坏，如块损坏、资料档案库完整性、健康检查等等.监控的时效性关乎企业从故障中可挽回的损失.比如由于没有提前做好恢复准备工作，2018 年 YouTube 宕机故障就曾导致超过 1 h 的服务中断.另一方面，要求在对数据库用户干扰最低的条件下保证信息的可用性.各类管理程序如果采样频率过于频繁，会与用户争夺资源，反而影响数据库表现的稳定性.因此，分布式数据库往往只保留与恢复工作密切相关的管理进程，比如检查点(checkpoint)、自动数据清理(auto-vacuum)机制等.

(2)自恢复.自恢复指数据库遇到问题时能够自动恢复到上一健康状态.首先，数据库从开始部署到提供服务的各个阶段都面临着各种软硬件隐患，如配置文件丢失；数据丢失、损坏、被恶意访问；服务器压力指数高时出现锁表、阻塞和大量慢查询等等.我们需要选择合适的恢复和备份方式解决相应问题.比如在 Oracle 的 RMAN 管理工具中，会对软硬件故障进行评级(低|高|严重)，对不同级别的故障推荐相应的恢复策略，如严重故障就要求恢复到过去时间点上的镜像.其次，自恢复问题往往与硬件直接相关，需要在不同形态的数据库(单机/集群/虚拟机)都有即时、精准的恢复能力.

(3)可视化的数据库管理工具.对于数据库管理员(DBA)来说，保持数据库运行在最佳状态需要对突发问题做出迅速准确的反应.单靠 DBA 人工从大量日志数据中统计出管理信息是非常困难的.数据库管理工具则能帮助简化数据库管理流程和日常维护任务.从自管理工作的需求出发，这类工具至少要具备四个方面的功能.① 提供各类查询结构信息.如 RDBMS 给出触发器、视图、外键和存储过程等信息；② 提供负载信息.如当前正在执行的任务信息、查询信息和算子信息等；③ 提供故障报告.将各类软硬件故障和处理方法以合适的方式展示给用户，并提供选择故障恢复策略的配置自由度；④ 提供元数据查询，有效的组织系统视图，可视化当前数据库状态，如分布式集群下各个节点的健康情况、元组读写数目和 IO 负荷等等.

1.2.8 数据库支持机器学习

从以上的数据库技术中可以看出，相同的数据库功能经常有多种方法或组件可供选择.例如数据备份分冷/热/温三种；存储层支持多种存储引擎；优化器提供多种计划选择策略；甚至针对不同的数据和负载可选不同类型的数据库等等.但无论是静态指定还是手工选择，对复杂多变的外界环境，我们都难以只通过统计分析做出准确选择.相反，另一种思路是将各个数据库方法乃至于整个数据库各自打包组成组件仓库，由一套决策机制统一根据负载和数据类型决定调用方法[15-16].

(1)机器学习作为用户定义函数(UDF).通过

把用户自定义程序加载到数据库系统，机器学习算法可以像内置函数一样由用户调用. UDF 在定制业务逻辑中为用户提供了很大的灵活性. 比如在机器学习中用 UDF 重载朴素贝叶斯、KNN 等分类算法，方便数据库处理相应问题.

(2) 机器学习作为物化视图. 数据库也可以抽象成一种类似马尔可夫的决策算法，其通过分析优化器在生成查询计划中的整个流程，了解各阶段信息的处理开销(估计器)，并预测未来可能发生的变化(缓存机制). 因此，我们可以用机器学习模型表示数据库的处理流程，利用模型的自主学习能力对整个流程进行优化.

(3) 自动机器学习. 现有数据库多是通用系统，不能为用户的特定工作负载和数据特性选择合适的算法. 比如对于范围查询，基于 B+树查找要远远好于其他传统索引. 但是，对于 B+树、倒排索引等传统数据结构，他们都是基于固定单一的规则存储数据，不能根据数据分布动态地调整局部结构. 因此，更进一步，我们可以用学习模型代替传统数据库组件，每次基于输入的数据(如离散、连续等特征)和输出期望(如分类、回归、评估等)选择合适的学习模型. 然后利用合适的策略(如 Adam、SGD 等)对预先训练好的模型做快速优化(fine-tune)，最后达到提供服务的水平. 基于这个流程，我们便可以从组件调度和操作执行两个层面实现一个基于学习的数据库，将机器学习自动化.

1.3　机器学习技术带来的挑战

机器学习技术给数据库发展带来新的机遇. 但是，由于机器学习算法对训练数据和时间有较高要求，而且数据库技术本身存在诸多问题，二者在结合过程中还面临着很多挑战.

1.3.1　高质量的训练数据少

机器学习算法多对训练数据有较高的要求，主要体现在训练样本质量、训练样本数目和训练样本多样性三个方面.

(1) 样本质量. 样本质量对于学习模型训练的效果有着最为直接的影响. 为了正确训练预测模型，需要样本数据是正确、去重的. 但是在数据库系统中我们很难保证数据质量. 首先，数据库系统虽然有丰富的统计信息，但是这些信息很多不是实时更新的，比如关系表的统计信息等. 其次，这些数据会受到很多因素的影响，比如在基数估计中，我们不仅需要考虑数据分布信息，而且要考虑数据库各类物理操作的执行代价.

(2) 样本数目. 光有高质量样本还不足以训练出健壮的机器学习模型. 以神经网络为例，如果训练样本很小，连接权重更新次数少，模型能够探索的空间也就非常有限，而且很可能出现过拟合的问题，缺少对新场景的适应能力. 但是很多数据库问题很难提供足够多的训练样本. 比如在负载预测中，真实场景的用户数据往往被严格保护，难以拿到. 如果不是用于企业内部优化，很难提供足够多的负载样本给模型训练.

(3) 样本多样性. 尽管成熟的机器学习模型具备适应新问题的能力，但是多样的训练样本可以加快模型的收敛速度，提高模型的准确性和适应性. 然而在数据库相关问题中，由于允许的训练时间有限，往往在最初训练阶段很难获得足够多样的数据，需要模型在较为单一的样本上先收敛到一个比较好的状态，之后在实际场景中，面对各种情况再逐步优化配置. 比如在调参问题中，训练阶段往往只提供几种典型的基准测试样本，需要学习模型有足够好的自优化和探索能力.

1.3.2　训练时间过长

光有数据对机器学习来说还不够，学习模型需要足够的训练时间来将较少的输入知识映射到输出知识. 因此，尽管原则上有更多的数据可以训练更复杂的分类器，而实际上往往简单的分类器才能被广泛使用. 因为训练复杂的分类器需要很长的时间. 同样的，现实场景下数据库系统面临各种用户需求、上线压力时，很难空闲足够长的时间等模型收敛. 因而数据库系统对机器学习算法的选择也有两个方面的要求. 其一，模型与业务场景适配. 比如调参问题中，很多参数都有连续的取值范围，再使用 Q-learning 等基于表方法的算法就要从极大规模的表中检索，浪费时间和资源. 其二，结合确定梯度下降(DDPG)、迁移学习(Transfer Learning)等提高训练效率的算法. 机器学习发展至今已经提出了很多优化训练效率的算法. 比如强化学习中 Actor-Critic 算法[17]，利用 Critic 对 Actor 的行为进行评估，可以加快整体模型的收敛速度.

1.3.3　适应能力不足

机器学习算法的适应性(泛化能力)多基于测试数据与训练数据符合相似的变化规律，如一个国家不同地区的房价等. 而当前数据库系统，面对多样的硬件环境、用户负载和用户需求，对机器学习算法的适应能力提出了更高的要求.

(1) 多样的硬件环境. 硬件环境的挑战体现在

两个方面. ① 数据库处理事务的硬件环境. 新的计算、存储介质不断涌现,如 SSD、磁盘阵列、NVM 等存储介质和 AMP、SMP 等多核 CPU 架构;② 针对不同的预算、应用和负载特性等,人们往往会为数据库配置不同的硬件环境. 而不同的硬件环境能对机器学习技术的实际表现产生很大的影响. 比如,调参技术中,很多参数与硬件强相关,如读/写元组的代价、数据库算子开销等等. 如果机器学习模型没有同时考虑硬件特性和查询特点,对于一个新的环境,模型需要花费较长时间纠正历史认知.

(2) 多样的用户负载. 用户负载多样性带来的挑战也体现在两个方面. 首先,针对不同的负载类型,数据库不仅需要提供合适的运维策略、执行计划、处理流程等,还要提供合适的架构存储数据. 如边缘侧(嵌入式、终端等场景),需要就近提供服务,存在数据/服务迁移等问题;管道侧,比如流式汇聚等场景,注重管道规划、有序性的维护,避免管道汇聚出现拥塞等;云侧,云数据库要对数据统一管理,维护分布式集群等. 这要求机器学习技术前期充分学习,过滤出负载中的重要因素,面对新的负载动态调整策略. 其次,混合业务的兴起增加了数据库管理的复杂度. 大量不同的负载类型往往对应着不同的服务,如银行系统中不仅要提供交易、转账等事务型业务,也需要进行用户数据审计、数据分析等分析型业务. 不同服务的融合导致数据库实例不能再仅仅为某一类负载提供服务,如 OLTP、OLAP 等,而要求能够支撑多种业务类型(如 HTAP). 这类复合业务对数据库的存储、处理和数据管理等各个方面都有了更高的要求.

(3) 多样的用户需求. 不同类型的用户往往对数据库的表现有不同的侧重. 例如实时性业务需要查询低延时;批处理业务要求较高的并行度、吞吐量等. 为了保证某一方面的性能,数据库常需要以牺牲其他方面的表现为代价. 例如,为了提升整体资源的利用率,可以增加数据库并行连接数,但是会导致各个连接分到的工作区减小,影响单个任务的执行效率. 在云数据库上,会存在各种用户“体验”的需求. 要使用机器学习优化数据库技术,需要机器学习模型能够自动、快速适应不同用户的需求. 并利用各种统计、缓存、数据整理机制提高资源利用率和容错防灾能力,进一步提升用户体验.

1.3.4 机器学习模型与数据库技术的匹配

数据库优化技术涉及到环境配置、查询优化和缓存机制等诸多方面,发展至今仍然有很多亟待解决的复杂问题. 比如,调参方面,数据库系统中有大量可调的系统参数,很多参数都有连续的取值空间,而且参数之间存在复杂的隐式关系,如何找到最优的系统配置是一个 NP-hard 问题;优化器方面,计划选择的状态空间关于表的数量是指数级的,在这样一个状态空间中找到最优计划也是一个 NP-hard 问题;而索引方面,传统索引如位图、基于树的索引等,由于本身数据结构的限制,无法有效检测“未知”用户行为和数据[12],没有把用户习惯与索引的选择关联起来. 这些问题虽然都有相关的机器学习技术可供使用,但真正将学习模型应用到实际场景下还存在着特征选取、反馈函数设计等诸多问题.

1.4 论文结构

为了帮助开发者和研究人员更好地把握当前基于机器学习的数据库优化技术的发展情况,我们选取五个研究较为充分而且具有现实意义的数据库研究方向做进一步综述:

(1) 第 2 节概要介绍数据库调参技术的研究情况以及存在的问题和挑战;

(2) 第 3 节概要介绍基数估计技术的研究情况以及存在的问题和挑战;

(3) 第 4 节概要介绍查询计划选择技术的研究情况以及存在的问题和挑战;

(4) 第 5 节概要介绍索引自动推荐技术的研究情况以及存在的问题和挑战;

(5) 第 6 节概要介绍物化视图自动选择技术的研究情况以及存在的问题和挑战;

(6) 第 7 节通过结合实际应用场景,进一步探讨数据库与机器学习结合的研究前景和发展趋势.

2 数据库调参技术的研究

数据库调参技术是指通过调整系统配置优化性能的一类技术. 数据库调参是一个 NP-hard 问题[4]. 合适的参数配置会大幅度提高系统性能. 例如,在 PostgreSQL 中,Work-Mem 和 Maintenance-Work-Mem 等参数控制系统的内存分配方式. 通过为它们设置适当的值,我们可以有效地利用内存. 随着 IoT 和云计算技术的快速演进和应用,对异质环境下快速进行自动化调参的需求日益迫切. 目前已经有很多数据库调参方面的研究成果[4,8-10]. 基于方法本身能否从历史任务中学习的特点,我们可以将这些方法分成传统的(非学习)调参方法和基于自学习的调参方法(见表 1).

表 1　数据库调参技术的相关工作

领域	方法		优点	不足	适用场景	表现	适应能力
数据库	专家决策		有经验保障；满足基本需求	手工配置耗时耗力；难与实际场景同步	单机，负载单一	低	低
	静态规则；统计学方法[18]		基于规则选择；易于实现	可应对的场景有限；只能调少量参数	单机，负载单一	低	低
	启发式搜索[4]		自动调参；调参速度快	难以保证在资源限制内找到合适配置；不能记忆历史经验	单机，分析型负载	低	低
	基于机器学习技术	特征抽取、传统机器学习[9]	自动调参；记忆训练经验	管道式架构；有特征损失；需要大量优质样本	单机，负载模式固定	中	高
		深度强化学习[10]	自动调参；效率高；小样本	只支持负载级别调参；没有考虑负载特征	云数据库	高	中
大数据分析引擎	随机抽样；图论[19]		存储开销较小；调参速度快	调参方案少；不能利用历史采样经验	单机，场景单一	低	低
	基于机器学习技术	递归搜索；传统机器学习[20]	不用实际跑负载测试；探索速度快	可调参数少；启发式算法，稳定性低	单机，分析型负载	中	低
		随机森林；深度学习[21]	调参效率高；基于多个网络的推荐结果，稳定性高	网络输入只考虑参数的默认值	单机/分布式数据库，负载模式固定	高	低

2.1　传统调参技术

在传统生产场景下，数据库调参单纯依赖于数据库管理员(DBA)．这些专家先通过手工反复实验得到统计数据，再根据经验决定配置方案．这里存在两个问题．首先，数据库一般具有高维参数空间，在有限资源下几乎不可能遍历整个空间来获得最优解．而通过对高维空间合理抽样来获得最优子空间对人来说是一件很困难的事情．此外，由于为每个参数样本重跑负载需要耗费较长时间，数据库专家往往需要数周的时间来最终得到合适的数据库配置．这不仅导致了大量的人力、资源开销，而且难以与动态变化的实际工作场景同步．因此有些人尝试用一些经典算法和概率模型来解决这些问题．

2.1.1　辅助决策的调参工具

当下有很多数据库调参的辅助工具．按方法大体可以分成两种：基于规则的调参工具和基于概率模型的调参工具．

(1)基于规则的调参工具．这类工具往往是由数据库厂商根据某一种数据库的特点设计出来．它们负责捕获数据库状态、表现和关键参数对数据库表现的影响程度，将结果呈现给用户，并给出参数调整的意见．比如 MySQLTuner①，它能够收集数据库的状态数据(如存储引擎信息、数据库活动等)．根据这些数据，会用固定的规则推荐一些修改参数的策略，例如“启用慢查询日志来排除错误查询”．这类工具存在两个问题．首先，基于规则的算法能够处理的场景是有限的．面对多变的数据库环境和复杂的参数配置情况，难以保持很好的表现．此外，这类工具一般不具有普适性．它们对调参对象的种类乃至版本号都有固定的限制，而且维护起来非常耗时耗力．

(2)基于经验模型的调参工具．另一种工具基于经验模型，通过从大量测试中总结“模糊规则”，发现数据库表现与配置间的相互关系，并通过可视化的方法动态呈现响应曲线，帮助专家直观地选择满足表现要求的配置．比如 Wei 等人[18]提出的一种基于模糊规则的调优工具，它主要包括三个阶段：其一，他们从自动负载仓库(Automatic Workload Repository，AWR)中提取调优相关的数据．AWR 是 Oracle 的一个表现测试工具，他们利用 AWR 获取负载、数据库状态、表现统计等信息．其二，通过在不同的配置下重复执行负载，他们观察表现量度与不同配置之间的关系，总结出调优相关的“模糊规则”．但是这些模糊规则不具有复用性．首先，在不同场景下需要重新统计．其次，每次统计都要从头开始，而不能基于历史经验进行．其三，基于得到的经验模型，调优工具可以为数据库管理员针对性地推荐参数．

2.1.2　基于启发式算法的自动调参系统

另一种办法则是利用启发式的算法自动化整个调参过程．Zhu 等人[4]提出了一个基于搜索的调参系统．其核心算法主要由两部分组成．首先，划分与分治取样(Divide and Diverge Sampling)把整个 N 维参数空间离散化：每个参数的取值范围划分成 k 段，每一段随机取一个值．这样整个参数空间便转化成由 k^N 个点组成的样本空间，为了保证抽样的多样

① http://mysqltuner.com/

性,要求每个参数值只能被取样一次.然后,有界递归搜索算法(Recursive Bound Search)每次在抽样空间中随机选取 k 个样本测试,得到表现最好的点 C_0.再以 C_0 为中心划分出下一次的抽样区域.这样迭代,直到不能得到表现更好的点或者达到资源限制.

以上方法主要体现了两类思路:(1)专家根据经验给出有限的调参规则;(2)抽样探索参数空间,在有限的资源下找到表现最好的区域.但问题是无论是人还是经典非学习算法,解决这类复杂的 NP 问题并不能达到令人满意的效果,往往只能得到固定的次优解.此外,它们都没能充分利用历史数据.也就是说,在每次重新启动调参过程后,都要从头开始,而不能基于之前训练得到的经验进一步优化模型.长期来看,这不仅会导致大量的资源和时间浪费,而且使得调参能力滞留在较低水平.

2.2 基于机器学习的调参技术

针对传统调参技术中的不足,越来越多的人转向机器学习领域.不同于基于规则的传统编程方式,成熟的机器学习技术从数据(训练样本)中学习输入与输出之间的依赖关系,从而能对未知的输入做出较为准确的预测.对于调参问题,由于很难给出一个场景下的最优配置,获取大量的标签数据变得非常棘手.所以选择合适的特征和学习模型变得十分重要.已有的研究成果中,人们主要使用了传统机器学习和深度强化学习两种技术.

2.2.1 基于传统机器学习的调参技术

传统机器学习包括线性回归、决策树等多种经典模型.其利用大量样本进行模式分类、回归分析和概率密度评估等.但这些模型在应用前往往要先进行复杂的特征工程,来获取更好的训练数据特征.

在 Dana 等人[10]基于大规模的机器学习方法提出的调参系统 OtterTune 中,他们就主要采用高斯过程(Gaussian Process)为当前负载推荐合适的参数.为了在探索(获得新知识)和开发(根据现有知识进行决策)之间权衡,高斯过程将参数与负载的关系视作多元高斯联合分布.任意数据集中 n 个观测量 $y=\{y_1,\cdots,y_n\}$ 都与一个随机变量关联,通过改变噪声水平决定样本是根据最优策略生成(开发)还是随机选择产生(探索).此外,在默认情况下,高斯过程提供了置信区间(confidence interval),即被测量参数测量值的可信程度,决定把哪些学到的知识持久化下来.整个调参过程可以分成特征抽取和参数学习两个部分.

(1)特征抽取.特征抽取是指从原始的输入/输出空间中,只选择与目标关系最紧密的一小部分特征真正用于模型的输入/输出.由于 OtterTune 主要根据当前负载的特性来推荐参数,提取负载特征来反映与参数之间的关联是非常重要的.OtterTune 从两个方面进行了特征选择:第一种是逻辑特征,包括查询语句和数据库 schema 的特征.另一种是数据库内部状态,即负载在执行时的状态变化量,如读/写页数和查询缓存利用率等.对于这些特征,OtterTune 首先利用因子分析(Factor Analysis)过滤无关特征,再利用简单的无监督学习方法 k-means,选择 K 个与参数关系最密切的特征,作为调参模型的真实输入.特征抽取只是帮助一些相对简单的算法来处理调参这样的复杂问题.往往过滤掉的特征中仍然具有大量的有用信息未被充分利用.

(2)参数学习.在抽取负载特征后,OtterTune 将当前负载 W 映射到一个最相似的负载模板 W'.然后运行高斯过程模型,生成一组与输入相关联的服从高斯分布的随机变量值,作为 W' 的最优参数推荐给数据库,用于执行 W.OtterTune 还会单独将 W 输入给调参模型,进一步优化模型.在训练阶段,OtterTune 会提高高斯过程的噪声值,增大探索未知参数空间的概率.

传统机器学习具有较强的泛化能力,使得这类调参系统在不同的数据库环境下都有较好的表现.此外,它能够有效利用在历史任务中学到的经验,应用在未来的调参工作中.但是,这种方法也有很多不足.首先,这类方法采用一种管道式架构,上一阶段获得的最优解并不能保证是下一阶段的最优解,而且不同阶段使用的模型未必可以很好地配合.其次,它需要大量高质量的样本用于训练模型,而这些样本是很难拿到的.比如,数据库表现受磁盘容量、CPU 状态、负载等很多因素的影响,很难去大量复现相似的场景.此外,仅仅使用高斯回归等模型很难优化数据库调参这种有高维连续空间的问题.这些问题也说明了引入功能更加强大的学习模型的必要性.

2.2.2 基于深度强化学习的调参技术

强化学习(RL)开始是为了解决博弈论问题而提出的.它本质上是提供了一种框架,使学习者(agent)能够在特定场景(environment)中采取行动,并按离散时间在与环境的交互中学习.与传统监督学习不同,RL 不需要大量的标注数据.相反,通过尝试和错误,agent 反复优化行为选择的策略,以最大化目

标函数.通过探索和开发机制,RL 可以在探索未知空间和开发现有知识之间做出权衡.但是,由于传统 RL 算法(如 Q-learning 等)都是基于离散的动作,如在 Q-learning 中还需要专门存储每个行为的评估值(Q-value),导致这类方法难以应对像调参这样有高维连续输出的问题.因而 Mnih 等人[19]提出了一种结合 RL 和深度学习方法的深度强化学习模型(Deep Reinforcement Learning),以解决先验知识有限、状态空间巨大的问题.

Zhang 等人[10]利用 DRL 方法提出了一种端到端的调参系统 CDBTune.它面向云数据库场景.在收到用户在线调优请求后,它从用户处采集工作负载,然后内部的 DRL 模型根据当前数据库状态推荐数据库参数,并在线下数据库中执行负载,记录当前状态和性能用于训练 DRL 模型.接下来,它使用通过离线训练获得的模型进行在线调整,最终推荐那些与在线调优中的最佳性能相对应的配置.如果优化过程终止,还需要再次更新 DRL 模型和内存池.

为了解决数据库调参这类有连续的输入/输出空间的问题,CDBTune 采用一种基于策略的深度强化学习方法:深度确定性策略梯度算法(Deep Deterministic Policy Gradient).不同于 DQN 等[22]其他 DRL 方法,DDPG 不需要为每个行为计算和存储 Q-value.它的 agent 包括 actor 和 critic 两个深度神经网络.首先,它从数据库中获取负载、参数、表现指标等数据,向量化成网络可以识别的张量.然后,在不同负载下,actor 可以根据当前环境的状态值直接给出合适的参数值,critic 对每次行为进行评估,对 actor 进行更新.最后,critic 基于每次推荐后的反馈值进行更新.

利用 DRL 模型高效的训练模式,这类调参技术可以极大地提升调参效率.首先,DRL 不需要大量的标签数据来训练网络.因为在一种负载下,就可以迭代产生多个训练样本.其次,融合 MDP、Bellman 和梯度下降等思想,网络能够迅速向目标拟合.

然而,CDBTune 也有一些不足之处.首先,它是负载级别的调参,而负载中不同语句之间往往会有不同的最优参数空间,导致对总体优化效果的制衡.此外,参数推荐只考虑了数据库状态,并没有直接结合负载特征进行分析,有一定的局限性.

因此,Li 等人[8]提出了一种基于深度强化学习的对查询感知的自动调参系统 QTune,从 3 个方面解决 CDBTune 中遗留的问题.其一,为了提高调参对不同负载的适应能力,QTune 在查询计划级别对负载特征(如读写比例、使用的关系表、执行代价等)进行编码,对执行开销做预估计.其二,为了进一步提高调参表现,QTune 采用了一种面向调参问题的深度强化学习算法(Double-State Deep Deterministic Policy Gradient,DS-DDPG),不仅基于 Actor-Critic 算法,提高训练效率;而且在强化学习的状态特征中综合负载、数据库状态指标、当前参数值等特征,提高调参策略的准确度和适应能力.其三,为了更好地满足不同用户对数据库性能的需求(如低延迟、高吞吐量等),QTune 增加 Query2Cluster 模块,先对所有用户负载根据参数偏好进行聚类,然后对相同类中的查询做批量调参和执行,从而平衡吞吐量和延迟两方面的需求.然而目前 QTune 仍然基于单机数据库,没能解决在分布式集群上为多个实例进行调参的问题.

2.3　面向大数据分析引擎的调参技术

大数据时代,除了需要数据库在分布式架构下面向不同的业务类型高速读写数据,提高分析引擎处理数据的能力也是一个重要的方向.从实现大数据并行计算的 Hadoop,到支持内存计算的 Spark,再到现在广泛用于大规模流处理的 Flink,分析引擎不断融合新的组件和功能来迎合不同场景的大数据处理需求.这也导致了分析引擎的内部配置更为复杂.调参工作显得尤为重要.

2.3.1　基于经典算法的大数据分析引擎调参技术

目前在分析引擎调参领域主要涉及图论、启发式算法和简单的机器学习模型等多种经典算法.Gounaris 等人[20]提出利用图解算法构建候选配置空间.该空间用图的形式表示:每个点表示 15 种固定的可选调参方案(每种方案下只修改 1～2 个参数),点之间的边表示方案的组合关系.算法每次首先对 15 种可选的调参方案随机组合,生成大量的配置样本.然后在不同样本下运行代表性应用,选择最好表现的参数组合添加到候选配置图中.该方法在已有经验(15 种基本方案)基础上探索更加复杂的调参模式来达到更好的效果.但它本身有很大的局限性.首先,整个配置图只有 15 个节点,相比于高维的参数空间(如 Spark 中有约 190 个可调参数)基数过于小.其次,采用随机抽样的方法不能利用历史经验,会造成大量没必要的时间和资源开销.

而 Nguyen 等人[20]则将参数推荐定义为一种搜索问题.他们首先使用拉丁超立方抽样(Latin Hyper-Cube Sampling)减小搜索空间.然后利用简单的机器学习方法(如 Decision Tree、SVR 等)训练出一个表现模型,其能够预估指定应用在推荐配置下的运行时间.最后,使用递归随机搜索方法

(Recursive Random Search),从样本空间中搜索最佳参数配置.虽然这种方法也先做了参数过滤,但是对保留的参数特征没有做范围限制,有利于探索表现更好的子空间.其次,它利用表现模型预测运行时间,避免了应用实际运行浪费的时间,从而大大提高了训练效率.

2.3.2 基于深度学习的大数据分析引擎调参技术

目前学习算法在分析引擎中的应用还处于起步阶段,相关研究主要使用深度神经网络模型完成调参任务.Gu 等人[21]提出了一种参数预测模型与表现模型相结合的方法:参数预测模型由 M 个神经网络组成.每次调参任务下,M 个网络根据默认配置推荐 M 套参数.表现模型是一个随机森林,它综合考虑应用、系统资源和集群信息,对这 M 套参数进行评估,选择最好的参数实际用于应用执行.这种方法利用神经网络强大的学习能力探索连续高维的参数空间与应用表现之间的映射关系.但其不足之处是,对于单个网络,输入仅有配置默认值,很难保证网络能够学到合理的映射模式.

针对 Spark 这类大数据分析引擎,以上方法并没能利用分析引擎不同于传统数据库的两点重要特性.首先,集群模式下,多个节点协同工作为一个应用服务.调参工作应该同时考虑集群特性,以总体表现作为优化目标.其次,分析引擎往往具有多个上层组件,例如 Spark 上的 Spark SQL、MLlib 等,导致 Spark 配置中不仅包括整个集群共用的公有参数,还有各个组件的私有参数.简单地将所有参数作为模型的输出,很难探索或学习出这两类参数的依赖关系.

2.4 未来方向

随着时代发展,主流业务场景已经从原始的单机作业向云端迁移.而且分布式的计算场景日益多样化,诸如集群计算、边缘计算等.数据库调参技术也不能单单考虑在单机数据库上的表现.相反,调参技术在多粒度服务和环境适应性方面的表现正在成为关注的热点[3].多粒度服务指的是调参技术不单单要能够以数据库作为调参对象,而且能够针对用户、负载或者单条查询进行动态调参.甚至于可以快速学习分析类查询的配置偏好,批量处理有相同配置偏好的查询.而环境适应性指的是调参技术能够快速适应用户、负载、硬件环境乃至数据库等多种级别的迁移,保持足够好的表现.

3 基数估计技术的研究

代价估计是数据库查询优化的重要组成部分,不论是枚举、动态规划还是启发式算法选择最优执行计划,都必须依赖于准确的计划估计代价,其重要性甚至大于执行计划选择算法本身.传统的数据库查询代价估计组件主要由两部分构成,基数估计器(cardinality estimator)和代价模型(cost model).基数估计作为传统数据库查询优化的重要环节已经被研究了数十年,但是目前最先进的用于生产环境中的基数估计器依然存在很大误差.误差甚至能够达到几个数量级.在如此大的基数估计误差的情况下,查询优化器无法准确估计执行代价,从而使得构建出来的执行计划无法达到很好的性能.根据 Leis 等人[23]在实际数据集上的测试,代价估计的挑战更多来自于基数估计的准确性问题.至于代价模型,也是在实际生产环境中被反复调整,部分系统运行的细节已经被细致考虑,比如在 PostgreSQL 中包含了超过 4000 行 C 语言代码来实现代价模型.即使这样,数据库查询代价估计仍然面临两种挑战:(1)设计一个更加准确健壮的基数估计器;(2)基于端到端架构实现自动代价估计.下面我们分别从传统基数估计和基于学习的基数估计两个方面对现有技术进行介绍(见表 2).

表 2 基数估计技术的相关工作

	方法	优点	不足	使用场景	表现	适应能力
基于统计的基数估计技术	直方图统计[24]	简单,被广泛使用	折衷估计误差和时空代价	被广泛使用	中	高
	数据画像,线性哈希算法[25-29]	实现对不同数据个数的估计	需要额外的位图空间和精心设计的哈希函数	单列数据估计	中	低
	基于索引的采样估计[30]	更加精确估计,面向连接查询的估计	需要额外的空间存放采样数据;存在 0-tuple 问题	内存数据库	中	中
基于学习的基数估计技术	对查询建模的监督学习方法[31]	目前最精确快速的基数估计	需要大量训练;泛化能力不足	离线训练,在线估计	高	低
	对执行计划建模的学习方法[32-35]	对查询的表达能力强;能够同时估计基数和代价	需要大量训练;泛化能力有限	离线训练,在线估计	高	中

3.1　传统基数估计技术

目前已有的传统数据库基数估计技术分为直方图(histogram)，数据画像(sketching)和采样(sampling)三类方法. 基于直方图的方法针对的是真实数据库中，每一列值的分布并不是均匀的，为了拟合数据的真实分布，维护一个直方图来保存每一个数据范围的信息. 基于画像的基数估计依赖于事先构建的数据表统计信息，集合基数估计常用的方法是使用一个哈希函数将数据映射到位图(bitmap)上，然后对位图进行基数估计，对这类算法的研究和改进包括MinCount[36]等. 对于多表连接查询的基数估计则采用经验公式，这种方式容易产生很大的基数估计偏差. 随着内存成本的下降，内存数据库兴起，基于采样(Sampling)的基数估计方法被广泛研究，基本的方法是将查询应用在从原始数据中采样出来的小数据集上来获得对整体分布的估计，这种方法在内存数据库 Hyper 中已经被应用.

3.1.1　基于直方图(Histogram)的算法

基于直方图的算法[24]根据边界$[b_1,b_2,\cdots,b_n]$将列数据划分成若干份，并且统计如下信息，一个是该属性位于b_{i-1}和b_i之间的元素行数，另一个是位于此范围不同的元素行数，这种直方图间隔也被称作是分桶. 直方图被分为两种类型，等宽直方图和等深直方图. 等宽直方图是指所有的分桶都具有同样的宽度，$b_i=b_{i-1}+w$，w 对于所有的 i 是一个定值. 等深的直方图指的是每个分桶中含有的元素行数是一致的(但是宽度可能不同). 等深的直方图相比于等宽而言更加适应倾斜的数据.

等宽直方图的桶宽计算方法如下，假设一列数据的最大值是 MAX，最小值是 MIN，那么桶宽就是$(MAX-MIN+1)/N$，其中 N 是桶的个数. 如果查询条件是一个相等查询，那么我们可以用直方图中存储的对应桶的行数除以桶的宽度来计算查询条件过滤之后的基数. 对于范围查询，叠加所有被查询范围完全覆盖的桶的行数，对于范围边缘的桶，用覆盖的宽度除以桶的宽度作为覆盖比，然后乘以桶的基数和之前叠加的行数相加得到最终的估计的基数.

等深直方图的深度计算方法如下，假设一列数据一共有$|R|$条元组，如果分成 N 个桶，那么每个桶中将近似包含$|R|/N$条元组. 等深直方图能够突出高频数据值的重要性，所以更加适应于倾斜分布的数据. 对于等值查询，估计的基数等于对应桶的总行数除以桶的宽度(桶的范围). 对于范围查找，范围中间桶的基数叠加，范围边缘的桶的基数按覆盖比例叠加.

直方图方法中桶的宽度(深度)和直方图占用的空间是一个折衷，宽度(深度)越小，那么数据刻画就越精细，能够估计的基数就更加准确，但是直方图占用的空间也更大，检索速度也更慢.

3.1.2　基于数据画像(Sketching)的统计方法

这类算法主要针对的是数据集合中不同元素个数的估计，实现这个估计最简单的方法是使用一个容量等于全局元素数量的位图(Bitmap). 这个位图初始化为全零，之后扫描全表，将遇到的所有的元素的对应位置标记为 1. 一遍扫描之后，只要计算位图中 1 的个数就能得出准确的不同元素个数. 这种方法估计精确. 但是需要的位图可能很大，而且全表扫描的代价也是不能接受的. 所以单纯使用位图的方法是不可行的，但是位图在现行的方法中依然扮演着举足轻重的角色. 使用排序(Sorting)取出重复元素也是传统的统计不同元素个数的方法，但是排序本身就是个昂贵的操作，所以对于当今的大数据集也是不可用的. 哈希法(Hashing)可以通过一次扫描对数据进行去重，这样比排序要快很多，但是十分消耗内存，为了减少碰撞，哈希表可能需要很大的空间而难以放入内存. 综合上面的方法，布隆过滤(Bloom Filter)结合了位图和哈希技术，维护一个保存哈希的位图，但是这种方法也需要事先知道元素个数的上界.

Tschorsch 等人[25]提出了一种 FM 算法，他们发现通过一个良好的哈希函数，可以将任意数据集映射成服从均匀分布的随机值序列，然后利用随机假设进行估计. 进一步地，他们计算了哈希值后缀中连续 0 的个数，发现在随机数集合中，如果哈希值的二进制后缀有 k 个连续的 0 值，则集合中大约有 2^k 个不同的元素. FM 的作者还指出使用随机平均(stochastic averaging)可以将每个元素的处理时间减少到 $O(1)$并且获得和 FM 同样的效果，这种算法叫做随机平均概率计数(PCSA)，该算法基本思想是尽可能将所有不同的元素分到 m 个桶中，然后通过估计每个桶的基数近似估计全局.

但 Alon 等人[27]指出 FM 算法基于理想的随机哈希函数假设是不现实的，取而代之的是，他们使用线性哈希算法，并且使用位图中前置的 0 的个数来估计不同元素个数，但是这种算法受到异常哈希值的影响较大，效果不如 FM 算法. LogLog 算法是 Durand 和 Flajolet 提出的一种基于 AMS 的算

法[28],它使用 PCSA 算法解决基数高估问题而且该算法只需要 $Log_2Log_2(N)$的内存. SuperLogLog[26]是 LogLog 算法的一个优化变种,这种算法降低了基数估计的方差并且减少了空间代价. HyperLogLog[26]类似于 LogLog 算法,但是它使用调和平均数来降低方差.

还有一类基于线性映射的方法,包括线性计数(LC)[27]和布隆过滤(Bloom Filter)[29]. 基本思想是使用线性哈希函数将数据均匀映射到位图上,根据位图上每个位置被访问到的次数,利用极大似然对基数进行估计. 后者为了在有限的空间中减少哈希结果碰撞,使用了多个独立的哈希函数,每个元素可以被映射到固定数量(少于函数个数)的位置上,布隆过滤的方法已经被广泛使用.

3.1.3 基于采样的方法

采样是一种能够替代基于统计法的基数估计方法. 它不依赖于特定假设,能够发现一些偶然的关联从而获得更加准确的估计. 尽管这种方法被研究了几个世纪,还是很少有系统将其用于生产环境,究其原因是磁盘输入输出代价过于昂贵. 但是现在随着内存价格的下降,出现了很多内存数据库,使得基于采样的基数估计更加实用. 简单的基于采样的方法就是从数据集中随机采样一小部分数据,将查询语句应用在小数据集上获得一些结果,然后根据采样比估计出整体的结果数目. 这种方式对于单数据集估计是有效的,但是对于多表连接操作则面临很多问题,比如有效采样可能会随着连接表数量的增加而快速减少从而使得采样效果低下. 针对这个问题,Leis 等人提出了一种基于索引的采样技术[30]. 这个技术基本思想是使用基于索引的相关采样替代每个表的独立采样. 独立采样会造成有效采样衰减的问题,所以对于 A 表和 B 表的连接,可以随机对 A 采样 N 条,但是采样 B 的时候,首先在 B 索引中扫描 A 的采样值的键,生成出所有 B 种和 A 采样有关联的元素,然后对这些元素进行采样,这样 A 和 B 连接之后的采样依然可以保证是 N 条而不会衰减. 这种方法的优点是效率高,准确性好,缺点是依赖于索引,不够灵活.

3.2 基于机器学习的基数估计

传统基于统计的基数估计的方法存在很大的误差,基于采样的方法在估计多表查询时存在有效采样衰减的问题,而基于索引的采样技术强依赖于索引结构. 近期的工作表明[31-32],利用机器学习可以实现高效、准确、可靠的基数估计方法. 使用机器学习算法进行基数估计大致可以分为两类,一类是只利用查询语句本身建模. 这种方法仅从查询语句本身抽取特征,不需要基数估计器访问数据表,也不需要知道查询的具体执行方式. 但是不能保证足够好的准确性和稳定性. 另一类是面向查询执行过程的建模,它基于查询优化器,支持执行计划级别的基数估计. 此外,它也可以直接对执行计划的代价进行学习,而不需要优化器的参与,从而节省了对代价模型参数进行调节的环节,使基数估计模型在特定环境中更加健壮,同时具有多环境的自动迁移能力.

3.2.1 面向查询的基数估计

传统线性的机器学习算法存在包括对高维数据抽象能力不足等诸多缺点. 最新的基于查询的基数估计算法[31]提出一个多集合卷积神经网络. 该网络由多个卷积神经网络堆叠而成,能够分别学习关系表数据、查询条件、连接条件之间的关系,从而学习一条查询语句的基数值. 它利用神经网络估计整个数据库中可能存在的查询的结果基数. 实验显示该算法模型估计出的结果基数远优于 PostgreSQL 以及基于采样的传统基数估计算法. 首先,利用深度学习技术,它可以高效地学习关系表数据、查询条件、连接条件之间的关系,比较高效地预测小查询. 其次,查询不用预先分类,特征也不仅仅局限于数值型的值,所有的查询中存在的信息都可以通过独热编码和数值归一化表示出来,对于涉及到表的数量较多的大查询也具有一定的泛化能力. 但实验结果显示,它对于复杂谓词的学习依然很困难. 此外,数据库查询多属于声明性语言(如 SQL),不包含具体的物理操作,所以单纯依赖查询进行基数估计有着严重的性能提升瓶颈.

3.2.2 面向执行计划的基数估计

研究人员试图进一步在执行计划中学习结果分布[32-35]. 由于查询计划树的构建过程是分步的,所有这类方案对于新一代的基于深度学习的优化器有很大吸引力. 最近的文章[32]提出使用强化学习的方式来学习,通过学习每个多表连接子树的表示获得一个通用的状态表示. 这种方法基于左深度(Left Deep)的查询计划构建方式,输入数据是经过独热编码的执行计划(每个节点单独编码). 模型中每个神经网络是一个单元,输入是已经执行的子查询计划的表示 h_t 和构建下一个查询计划所加入的新动作 a_t,输出是新的查询计划的表示 h_{t+1},使用一层全连接神经网络表示和学习这个映射 $NN_{ST}:(h_t,a_t)\rightarrow h_{t+1}$. 除了状态转换映射 NN_{ST} 之外还需要学习一个观测

网络 $NN_{observed}$，利用每层对应的真实基数值，可以很好地训练这两层网络.这种方式的优势在于网络可以在不同长度的执行计划中被反复调整直到适应所有的子查询，根据计划特性提供更精准的基数估计.此外，通过解析执行计划，它对于不同类型查询的适应能力也大大提升.劣势在于目前使用的神经网络模型在一些场景下不适用，如不支持灌木丛(bushy)的计划构建方式.所以为了支持不同结构的执行计划，Sun 等人[35]提出一种基于树形长短期记忆网络的基数估计框架.首先，由于常见查询计划都是二叉树结构，他们用一种树形模型来表示子查询计划，每个节点是一个 LSTM 单元，通过传递中间节点特征，可以同时学习查询基数和开销.其次，他们在特征提取中同时考虑了查询特征(如过滤条件、关系表)和物理操作(如算子、连接顺序).我们将这些特性嵌入到我们的树结构模型中，并利用一种有效的字符串编码方法，提高谓词匹配的泛化能力.

3.3　未来工作

之后的基数估计研究将着力于构建一个高效、健壮、可用的面向查询优化的代价估计器.主要是基于树形结构的模型，对查询执行计划进行特征工程，包括考虑如何对谓词表达式，节点操作进行编码，其中谓词表达式本身也是一个树形结构，而且表达式种类很多，包括连接条件，过滤条件，条件中的值存在数值类型，字符串类型等.复合表达式也包括多种连接符和逻辑符号等，所以表达式的估计是当前基数和代价估计的难点.目前工作将数据的 bitmap 加入特征向量中用于辅助学习，之后将寻找一种能够不用借助采样就能够独立学习的模型.

此外，目前的基数估计和代价估计是两个不同的阶段，这两个阶段都需要人去调整参数，存在较大误差.后面的工作将寻找一种模型能够同时解决这两个相关问题，让基数和代价能够互相帮助和调整，最终为查询优化器提供高效可靠的优化依据.

4　查询计划选择技术的研究

数据库的计划选择器一直是数据库领域一个重要的研究问题.对于一个查询语句，数据库的计划选择器会生成相应的连接计划.不同的连接计划对于查询效率有非常大的影响.SQL 的连接计划一般以树的形式(Join Tree)给出，它的状态空间的大小关于表的连接数量是指数级的.在这样的一个状态空间中，找到最好的计划是一个 NP-hard 的问题.以数据库的估计器为衡量指标，不同的计划选择器算法，通过不同的方法在状态空间中进行枚举，最后选择估计代价最低的计划.目前已经有许多的计划选择算法被提出，根据算法选择连接顺序使用的基准不同，我们将算法分为传统的静态连接计划选择算法和自适应性的 Join 计划选择算法(见表 3).

表 3　查询计划选择技术的相关工作

	方法	优点	不足	适用场景	表现	适应能力
静态的计划选择	动态规划[37]，右深树，zig-zag，左深树	能给出理论上该搜索空间最好的计划	计算开销大；选择时间长	有准确的代价估计器，且关系表数量不多	中	低
	随机采样，启发式搜索[38]	节约时间	不能保证计划质量	有准确的代价估计器，且关系表数量不多	低	高
基于学习的计划选择	选择度估计，深度学习[39]	根据执行结果调整估计器，提高代价器的准确度	准确度受估计器限制；要存储历史计划，增加存储开销	在估计器估计不准时准确，采集运行结果进行微调	中	中
	强化学习[11,40-42]	计划的性能不受限于估计器；提高选择效率；直接以运行时间为反馈，更加精准	传统 RL 方法泛化能力弱；现有深度强化学习方法的模型表征能力有待提高	关系表的数量较多；没有精准的代价模型	高	中

4.1　静态计划选择算法

连接计划选择算法在给定一个 SQL 时决定执行计划的连接计划，不同的连接计划会产生不同的代价.静态的连接计划选择算法使用确定的估计器作为连接计划的评估指标.这个估计器往往是人们根据经验和实际情况设定的，涉及到对结果的基数估计，CPU 和磁盘的访问代价估计等等.静态的连接计划选择算法在对状态空间中的可能计划进行探索，选取估计器上表现最好的计划作为输出.一个 SQL 可能的连接计划的状态空间数量是巨大的，完全的枚举所有可能的计划是不可行的.不同的做法使用了不同的搜索策略，从而对计划进行有效的搜索.根据搜索策略的不同，我们将其分为了基于动态规划的做法和基于启发式搜索的方法.

4.1.1　基于动态规划的计划搜索方法

基于动态规划的搜索方法的核心思想是通过对冗余状态的合并，减小搜索空间的大小.它已经被用在 PostgreSQL 中[37].在搜索连接计划的时候，对于

把同样的几个表连接起来的计划，我们只选取其中代价最小的计划. 比如 $A\bowtie B\bowtie C\bowtie D$，对于子计划 $A\bowtie B\bowtie C$，我们有两种不同的连接计划：$(A\bowtie B)\bowtie C$ 和 $A\bowtie(B\bowtie C)$，代价分别为 10 和 20. 我们保留 $(A\bowtie B)\bowtie C$ 作为子计划 $A\bowtie B\bowtie C$ 的最佳计划. 当考虑 $(A\bowtie B\bowtie C)\bowtie D$，使用 $A\bowtie B\bowtie C$ 的连接计划 $(A\bowtie B)\bowtie C$ 并连接得到 $((A\bowtie B)\bowtie C)\bowtie D$ 为该状态的结果. 即我们考虑 n 个表 $S=(T_1,T_2,\cdots,T_n)$ 的最佳 Join 计划 $Best(S)$ 时，我们枚举它的一个子集 $S_1=(T'_1,T'_2,\cdots,T'_m)$，$S_2=S-S_1$，我们使用 $Best(S_1)$ 和 $Best(S_2)$ 进行连接，得到 $Best(S_1)\bowtie Best(S_2)$ 作为 $Best(S)$ 的一个候选计划. 我们按照集合的大小为顺序，对 $Best$ 函数进行求解，对于一个集合 S，枚举可能的左表集合 S_1 和右表集合 S_2. 容易知道对于 n 个表 $S=(T_1,T_2,\cdots,T_n)$ 的连接来讲，它有 2^n 种状态，同时该方法的计算代价为 $O(3^n)$.

动态规划的做法仅仅是合并了冗余的状态，去掉了明显不可能成为最优的连接计划，他仍然是在全部状态空间上的搜索. 动态规划通过遍历状态空间能够给出估计器上最好的计划. 虽然减少了搜索空间，但动态规划的计算量仍然过大，从而影响了运行时间. 一些做法通过限制动态规划搜索树的形状来降低运行时间. 比如 left-deep 通过限制每次连接两表的右表为一个简单表. 即计算 n 个表的 Join 时，$S_2=T_x$，$S_1=S-(S_2)$. 该方法能够得到所有左深结构（右孩子都为叶子节点）的 Join Tree 的计划中最好的计划. 与之类似的还有 Right-deep（连接的左表为简单表，右深的树结构），zig-zag（左表或者右表，其中之一为简单表），这三种做法都能够在 $O(2^n)$ 的时间内得到该结构下估计器上最好的计划.

4.1.2 基于启发式算法的连接计划选择算法

基于启发式的连接计划算法，通过一定的搜索策略对部分的状态空间进行启发式的搜索，对比动态规划的遍历的做法它能够节约大量的时间，但是不一定能够保证得到最好的计划.

基于随机的搜索算法具有一定的随机性，它通过随机函数在状态空间中进行探索. 最基础的做法是 QuickPick-1000[38]，它通过随机地生成 1000 个可行的连接计划，利用估计器进行评估得到这 1000 中最好的计划作为输出. QuickPick-1000 能够很快地给出一个连接计划，同时由于它没有对搜索空间进行限制，可能给出很差的计划. 一些做法尝试使用启发式的随机信息来指导计划的生成. 基因优化器[37]的做法是一个优化的基于随机的做法，它同样被采用于 PostgreSQL 中. 基因优化器主要用于解析计划形式为左深（left-deep）的 Join Tree 结构. 对于一棵左深的 Join Tree，我们可以使用一个序列来表示. 比如 $((A\bowtie B)\bowtie C)\bowtie D$ 可以被表示成 $A-B-C-D$. 与求解 TSP 问题类似，基因优化器首先生成 m 个随机的序列来表示初始的 m 个连接计划 P_i. 在之后的计划求解中，我们通过采样从目前的计划中选取下一次迭代更新使用的计划. 使用评估器衡量每个计划的代价 $cost(P_i)$，代价越低的计划，拥有更高的被采样到的概率，比如设定 $1/cost(P_i)$ 为采样的权重. 在每一步的迭代中，我们选取两个采样到的计划进行交叉匹配，从而生成新的计划. 我们可以使用不同的策略实现两个计划的交叉. 一个可行的方法是对于序列 13524 和序列 12345，我们选定位置 2，并将 2 之前的序列进行交叉得到 12524，13345，对于两个序列位置 2 后重复出现的“2”和“3”. 我们将它们顺序替换得到 12534，13245. 由于采样的概率随着计划的代价改变，我们期望随着迭代次数的变多，每一步候选的计划越来越好. 我们将所有生成计划中代价最低的作为最后的计划返回. 爬山法[37]通过随机产生了一个连接计划，它每次针对一个计划进行调整得到下一个计划. 我们同样针对 left-deep 形式的状态进行求解，定义一个 left-deep 的计划为一个序列. 一个计划的邻接计划定义为随机交换序列中的两个位置后产生的计划. 我们选取一个计划邻接计划中最小的计划作为后续计划，循环直到后续计划大于当前计划，一次探索停止. 通过多次随机初始计划，完成多次探索，选取最好的计划进行返回.

基于贪心做法的搜索搜索算法，根据贪心的思路，设定一个局部评价标准，不断选取目前衡量下最优的计划. 是一种基于经验设计的算法，搜索状态空间中可能成为最优结果那部分计划. GOO 是一种贪心的策略. 对于 N 个表 $S=(T_1,T_2,\cdots,T_n)$ 的连接，当前局面下我们有 n_2 种可能的 Join 操作，我们选取两个表连接完成后产生的临时表最小的 Join 操作作为选择. 连接完成后我们得到 $n-1$ 个表，比如选取 T_1 和 T_2，则 $S'=(T_1\bowtie T_2,T_3,\cdots,T_n)$. 重复这一过程直到 N 个表都连接起来. 每次连接两个表的复杂度为 $O(n^2)$，整个算法的复杂度为 $O(n^3)$.

4.2　基于机器学习的计划选择算法

静态的连接计划选择算法依赖于估计器的性能.然而一个估计器上代价最小的计划在真实的运行过程中不一定是运行时间最小的计划.因此静态的连接计划选择算法的性能受限于估计器的效果.而人工定义的估计器很难覆盖所有的情况,做出准确的估计.因此研究者们使用执行时间作为反馈,来指导选取更好的计划.根据反馈指导的方式不同,我们将他们分为了基于估计器的适应性做法[39]和基于强化学习的适应性做法[40-42].

4.2.1　基于估计器的自适应性方法

这类做法主要认为连接计划的质量由估计器限制,他们在执行计划的时候尝试去更新估计器的参数,使估计器能更好地和真实运行的时间吻合.

一种实现方法是提高估计器在选择度(Selectivity)上的估计.选择度是一个很重要的指标,用于表示数据库一列上不同取值的数目估计.适应性的选择度估计做法[39]的思路是在数据库的运行过程中,根据每个列得到的反馈进行建模,而对未来该列的选择度估计提供帮助.考虑我们想到得到一个数值型列在$[l,r]$上的选择度 s 估计,我们考虑这个列的数据分布为函数 $f(x)$.该选择度为 $f(x)$在$[l,r]$上的积分,假定 $f(x)$的原函数为 $F(x)$,则 $s=F(l)-F(r)$.我们把 $F(x)$考虑成 n 个提前选择函数的加权和 $F(x)=a_1G_1(x)+a_2G_2(x)+\cdots+a_nG_n(x)$.则 $s=a_1(G_1(r)-G_1(l))+a_2(G_2(r)-G_2(l))+\cdots+a_n(G_n(r)-G_n(l))$,$a_i$为需要学习的参数.当一个 sql 执行后我们可以得到区间$[l_i,r_i]$真实的选择度 s_i'和估计的选择度 s_i.我们通过最小化误差$(s_1-s_1')^2+(s_2-s_2')^2+\cdots$,来更新参数 a_i.从而得到对于选择度更好的估计,并选取出更好的计划.

还有一类做法是结合计划对估计器的参数进行调整,像 LEO[38].LEO 对于执行过的计划,他能够得到计划每一步的代价.连接计划是树结构(Join Tree).LEO 将执行过的计划通过哈希(hash)的方式存放到哈希表中.在之后的学习过程中,对于一个计划,它通过递归的方式对树进行遍历,并在哈希表中查找相应的计划.找到匹配的计划之后,与目前的估计进行对比,用于更新估计器的参数,并指导计划的生成.

4.2.2　基于强化学习的适应性算法

强化学习(Reinforcement learning)的提出主要是用于控制理论.针对一个 RL 问题,它由几个部分组成:动作(actions)空间用于表示这个问题中系统可以采取的操作.状态(state)空间用于表示系统中所有的状态.智能体(agent)用于表示系统的智能单元,常常包括一个策略函数(policy function),策略函数以状态为输入,操作为输出.奖励(Reward)用于表示智能体采取一个操作时得到的奖励.环境(environment)用于与智能体交互,当智能体采取操作时返回新的状态,并给出奖励.

Tzoumas 等人[11]提出了基于 RL 的优化器做法.首先介绍了如何将 RL 中的概念和 Join 问题进行对应.将所有的 Join 条件作为动作空间.状态空间由执行 Join 过程中的所有连接树(Join Tree)组成,每一步执行的代价作为奖励,数据管理系统为交互的环境.我们的目标就是求解策略函数,即针对每个状态下如何采取操作.策略函数是 RL 的重要部分,通过定义长时奖励来求解它.一个长时奖励表示从一个状态到终止状态每一步 reward 的总和,它表明了一个状态的全局的一个估计值,策略函数通过最优化每一步的长时奖励得到.由于我们每一步使用的是单步 Join 的代价作为 reward,则长时奖励在问题中就对应到这个 Join 计划的未来的代价.一旦我们的策略函数能够最小化长时奖励,根据策略函数我们就能得到最小代价的 Join 计划.

在这个问题中我们以系统的运行时间作为指标,基于强化学习的连接选择器就可以不断地以运行时间作为反馈,算法给出的计划的性能便不再受限于估计器的性能.

有许多的 RL 算法被提出用于解决最小化总的代价,给出策略函数.其中 Tzoumas 等人[11]使用 Q-learning 完成了策略函数的求解.Q-learning 是一种基础 RL 的做法.对于一个给定的状态,定义 Q 值表示这个状态的长时奖励.Q-learning 使用一个表格 Q-table 记录下每个状态的 Q 值.为了更新一个状态 s 的 Q 值,通过操作 a_i我们能够得到后续状态 S_i,在所有的后续状态 S_i中 $Q(S_i)+reward(a_i)$最小的那个作为我们选择的状态我们用来更新 $Q(s)$.同时 $Policy(S)=a_i$.算法的每一步需要检索最多 n^2个动作,因此它的复杂度为 $O(n^3)$.与 GOO 有着相同的复杂度和相似的思路,不同的是,Q-learning 根据一个从目前到结果的最小的代价位标准而 GOO 仅仅选取当前 Join 表的大小为标准.同时 Q-learning 能够根据运行的时间作为反馈来对 Q 值进行指导

学习.

然而传统的强化学习方法存在一些问题.首先,Q-learning 需要使用一个表格 Q-table 记录下所有的状态.这个表格的大小很大,同时 Q-learning 无法很好地处理没有出现过的状态.近期的工作为了解决这一问题,提出了基于深度强化学习(DRL)的做法,像 DQ[40] 和 ReJOIN[41].与传统的强化学习做法不同,深度强化学习做法使用神经网络来对问题的状态、操作等进行表示.考虑 Q-learning 的深度学习做法深度 Q 网络算法(DQN),它的核心思想是取消 Q-table,使用一个网络 Q-network 来代替.对于一个状态 s,之前的做法是在 Q-table 里面进行检索状态 s(Q-table 中的数据是五元组〈当前状态,行为,下一状态,反馈值,终止标志位〉,在训练中自动生成),现在使用 Q-network 来对 s 进行计算 Q-network(s).同时对于一个从执行器得到的反馈用状态 s 和长时奖励 Q 表示,记作(s,Q).我们将 Q 作为 Q-network 中 s 的目标对网络进行学习.为了能让网络对状态 s 进行计算,我们需要先将状态 s 表示成一个向量,比如使用一个独热编码:1 和 0 分别来表示 Join Tree 中每个点是否在树中,同时对于每个列,使用选择度作为这个列的特征,从而构建出这个状态(Join Tree)的表示向量.得益于神经网络强大的学习能力和泛化能力,它能够记录下曾经学习过的 Q 值,并且对于没有见到的状态 s',也可以给出一个很好的预测.结合了神经网络的 DRL 模型在连接计划生成中得到了很好的结果.其次,以上方法中使用的特征向量都是定长,需要人利用经验预先设定好,这可能导致一些树的结构信息丢失(如相同的子树结构可能有相同的特征向量),进而产生较差的连接计划.所以 Yu 等人[42] 提出了一种结合树形长短期记忆网络(Long Short-Term Memory,LSTM)和强化学习的优化器 RTOS.RTOS 主要在两个方面对现有的 DRL 方法进行改进:(1)RTOS 采用图神经网络捕获连接树的结构,提高对不同结构的识别能力和优化程度;(2)很好地支持了数据库模式和多别名表名的修改,提高连接决策的适应能力.

4.3 未来工作

结合神经网络的深度强化学习算法被证明可以在连接计划问题中取得很好的结果.同时利用深度强化学习从运行时间为反馈对模型进行调节,使得模型性能不受限于估计器的性能.基于这个深度强化学习的解决思路我们可以在三个方向继续工作使得基于深度强化学习的连接计划算法得到进一步的应用.

(1)更强大的数据表征方法.现在的工作使用了简单的向量作为表示中间连接状态的工具.然而我们知道中间的连接状态是一个树的结构.平面向量很难表征这个结构.图神经网络(GNN)在这几年兴起,由于它对图结构的表示能力,被广泛地应用于复杂结构的事物的学习中,比如社交网络和知识图谱.我们可以使用图神经网络来对 Join Tree 进行建模获得更精确的表示.

(2)深度 Q 网络算法的特点是直接学习评判策略.但模型收敛需要网络一次次根据真实反馈值进行参数调整,整个时间成本很高.因此,可以采用类似于 actor-critic 的策略,用策略网络和评估网络联合交叉训练的方式加速训练过程,减少与外界真实交互花费的时间和资源.

(3)动态特性的支持.数据库元数据是会改变的,当插入表和增加列的操作被执行时.数据库的结构发生改变.而目前的神经网络设计依赖于固定长度的特征向量表示.不能很好地支持这些动态的操作.

(4)迁移算法.神经网络是一个基于学习的做法,当一个数据库构建时,它不能很快地开始工作,而需要进行一定时间的训练.而传统的连接计划选择算法可以在任意时间开始工作.一个可行的思路是利用迁移学习.在云上数据库等情境下,我们有着许多的数据库,不同数据库的列之间可能存在着相近的分布,我们利用这些分布信息.当一个新的数据库生成时,我们可以用其他数据库的网络信息对这个数据库的网络信息做初始化等等,从而减少训练代价和训练时间.

5 索引自动推荐技术的研究

索引对于提升复杂数据集上检索任务的效率有着非常重要的意义.不同的索引技术和索引选择算法对于数据库处理数据的效率有较大影响.因此,基于索引推荐的不同阶段,下面我们分别从索引生成和索引选择两个方面对相关技术进行介绍(见表 4).

表 4　索引自动推荐技术的相关工作

	方法	优点	不足	适用场景	表现	适应能力
索引生成技术	范围索引，分层递归[43]	减少空间占用；降低时延	“最后一英里搜索”问题	静态数据	中	低
	哈希模型索引[43]	减少冲突	表现与哈希映射体系结构强相关	数据分布固定	中	中
	基于学习的布隆过滤[43-44]	提高空间利用率	需要标准布隆过滤器辅助监督	数据分布固定	中	中
索引推荐技术	在线索引推荐[45-46]	针对工作负载变化及时调整索引；减轻 DBA 的工作负担	没有充分利用 DBA 经验；持续运行占用系统资源；连续调整，有一定盲目性	单机，数据动态变化	中	中
	离线索引推荐[46-47]	能够充分利用DBA的反馈	无法处理工作负载的动态变化；加重 DBA 的负担	数据变化小	低	低
	半自动化索引推荐[48]	充分利用调优器的计算能力；减轻 DBA 的负担；充分利用 DBA 的专业知识	没有考虑一个工作方案对整体的影响	数据动态变化	高	中
	分类器，强化学习，遗传算法[49]	能够适应工作负载的变化；推荐策略能够动态变化	主要利用遗传算法，没有深入利用机器学习算法	数据动态变化	高	高

5.1　索引生成技术

索引对于提升复杂数据集上检索任务的效率有着非常重要的意义. 因此，需要有效的索引技术来支持数据库操作. 传统或非人工智能索引方法，如位图索引、基于树的索引等，在数量、速度、变化、变异性、价值和复杂性方面能够满足需求，但它们无法对“未知”用户行为和大数据有效检测. DBMS 中常用的索引都是通用的数据结构，它们没有对数据的分布进行分析和利用，也没有利用现实世界中更流行更普遍的模型. 此外，数据集的增长也会导致索引大小的增长. 因此下面我们分别综述三种学习型索引.

5.1.1　学习型索引

(1) 范围索引. 范围索引是在一个有序集合中将查找键映射为记录位置. Kraska 等人[43] 提出了一个递归模型索引(RMI)，通过预测给定排序数组内键的位置有效地近似于累积分布函数(CDF). 使用学习模型替换 B-树的一个最大挑战是“最后一英里搜索”. 比如，使用单个模型将预测错误率从 100 M 减小到数百数量级往往很困难. 然而，将错误率从 100 M 减到 10 k 却简单得多. 因此，Kraska 等人[43] 提出了一个层次模型，在每个阶段，模型将键作为输入，并基于它选择另一个模型，直到最后阶段，模型的输出为记录地位置. 这样，与 B-树的结点类似，每个模型负责键空间的某个区域，以更低的错误率进行更好的预测. 另外，递归模型也可以根据需求，实现不同学习模型的混合使用，比如顶层模型可以选择 ReLU 以学习更大范围的数据分布，而底层则使用时间和空间代价更小的线性回归模型.

(2) 点索引. 哈希映射索引，其关键问题在于尽可能避免冲突，冲突会对性能或存储要求产生严重影响，而机器学习模型可能会提供减少冲突数量的替代方案. Kraska 等人[43] 提出了哈希模型索引，同样使用递归模型的结构，但是如何处理插入、查找和冲突取决于哈希映射体系结构. 学习键分布的 CDF 是学习更好的哈希函数的一种可能方法，可以通过哈希映射的目标大小 M 来扩展 CDF.

(3) 布隆过滤器. 布隆过滤器是一种用于测试元素是否是集合成员的概率性数据结构，有很高的空间利用率. 布隆过滤器使用一个 m 大小的数组和 k 个哈希函数，每个哈希函数将键映射到数组中的某一位. 虽然布隆过滤器空间利用率很高，但是对于大数据集，依然会占用很大空间. Kraska 等人[43] 提出了两种通过机器学习方法来建立布隆过滤器的思路. 第一种构建布隆过滤器的方式是作为二元概率分类任务，即学习一个模型 f，它可以预测查询 x 是键还是非键. 模型的输出为 x 是键的概率，那么，选择一个阈值 τ，输出大于 τ 就认为 x 是键，这样就可以把模型转化为布隆过滤器. 第二种构建方法是学习一个哈希函数，使得键之间的冲突和非键之间的冲突最大化，而键和非键之间的冲突最小化. Mitzenmacher 等人[44] 给出了一个更正式的学习型布隆过滤器模型，使用一个神经网络来预测一个值在集合中的概率，并用一个小的标准 Bloom Filter 来避免漏报，神经网络输出大于 τ，则该值在集合中，否则不在.

5.1.2　未来工作

Kraska 等人[43] 开创性地使用人工智能方法创建索引，开创了一个全新的研究方向. 然而，文章中

的模型大多都是初步想法，缺乏可靠的理论支持，也没有应用于解决实际问题，仍然有大量问题需要进行研究，如：

(1) 目前已有大量的机器学习模型，可以尝试将不同的模型应用于索引技术中；

(2) Kraska 等人并未考虑多维索引的问题；

(3) 从理论上证明学习型索引的正确性.

5.2 索引选择技术

在数据库系统中，在一个数据库的多张表上建立索引存在很多种索引方案，每种方案带来的收益以及对其进行维护的代价差别很大，因此，选择一个最优的索引方案成为一个重要问题，很多研究者对索引选择问题进行了理论研究，最终证明索引选择问题是一个 NP 完全问题[49]. 此外，也有很多研究者致力于解决实际问题，将索引选择问题应用到不同的场景中，开发了很多原型系统，提出了很多优化的解法[49]. 下面我们分别综述四类索引选择技术：(1) 利用反馈机制，不需要 DBA 干预的在线方法[45-46]；(2) 有 DBA 参与的离线方法；(3) 半自动化的索引选择；(4) 基于机器学习的索引选择(见表 4).

5.2.1 在线方法

在线方法的工作流程大致可以分为工作负载分析、索引方案选择、索引方案实现这三个主要步骤，这三个步骤的循环进行，使得在线方法能对工作负载的动态变化进行监测并及时做出响应. Lühring 等人[45]给出了一个基于“观察-预测-反应”循环的软索引自治管理方法，支持在现有预调整配置之上自动在线实现索引结构，自动调整框架本身的性能.

(1) 工作负载分析. 在线方法对工作负载进行实时监控，对工作负载进行分析. 它先对工作负载单个查询进行分析. 对于单个查询，通过对其中的 SELECT、WHERE、ORDER BY 和 GROUP BY 子句进行分析，提取可索引列，然后采用与 DB2 Advisor[46] 相同的方法枚举候选索引方案，并对索引方案的空间大小进行估计. 每个索引的收益，采用该索引被使用后查询执行时间的减少量来度量. 对于工作负载的分析，只需要将对单个查询的分析结果合并即可.

(2) 索引方案选择. 在对工作负载进行分析后，根据所得到的统计信息，利用各种求解优化问题的算法进行求解. Lühring 等人[45]采用简单的贪心策略来解决该问题. 首先，将候选索引方案按照相对收益(索引收益/索引大小)对所有索引方案进行排序. 然后，在保证空间约束的条件下，采用贪心策略每次选择候选方案放入结果集中，直到无法容纳其他方案，新的索引方案产生. 当新索引方案的收益与原有方案收益比超过某个阈值时，才使用新方案进行替换. COLT 则通过将索引选择问题建模为背包问题，给出新的索引方案. COLT 会计算每一个索引的净收益，即会从上一阶段得到的收益中，扣除由于索引创建、删除和更新等带来的负面影响. 根据每个索引的大小和其净收益，将空间约束作为背包大小，索引视为物品，按照背包问题的动态规划算法，就可以得到一个索引方案. 但是，简单应用背包问题进行求解没有考虑索引之间的依赖性，需要对结果进行调整，从中删除一些因为依赖关系而导致无用的索引.

(3) 索引方案实现. 当新的索引方案生成后，需要根据索引方案进行实现，创建新索引或对已有索引执行更新或删除操作. 这一步骤理论上可以在任何时间执行，可以在索引方案生成时立即执行，也可以选择在系统空闲或者低负载的时间运行. Lühring 等人[45]将索引实现过程延后进行，即在系统空闲时间执行. 为了索引效果能够尽快体现，Lühring 等人在反应期引入了两个操作 IndexBuildScan 和 SwitchPlan. IndexBuildScan 对通常的表扫描算法进行了扩展，将要生成的索引集合作为一个额外的输入，在扫描的同时建立索引. SwitchPlan 允许在执行与创建索引的查询再次执行时，使用最新创建的索引. 例如，对于嵌套循环连接，内层循环地表会被扫描多次，第二次扫描时，就可以利用先前建好的索引. COLT 则选择在索引方案生成后立即进行实现，在索引方案生成的同时立即发出异步请求进行索引实现，以尽快对新索引进行利用.

Lühring 等人[45]的方法和 COLT 考虑的方面不同，前者主要考虑对系统性能的影响，后者则希望索引的作用能够尽快被利用. 如果工作负载连续不断变化，系统始终处于高负载状态，他们选择的策略会导致推荐的索引配置还未实现，工作负载的变化就已经触发了下一次索引选择过程，索引方案的效益不仅不能体现，还会因为频繁的优化过程占用大量系统资源.

5.2.2 离线方法

离线方法往往需要数据库管理员的控制，不需要对系统进行监视. 对于一个工作负载，离线方法往往会以单个查询作为起点进行分析处理，然后进行合并和优化，最终得到一个优化的索引集合. Elghandour 等人[46]将索引优化问题建模为背包问

题的变种，提出了一种充分利用 DB2 自身优化器进行索引推荐和代价评估工具 DB2 Advisor[46]. 而 Chaudhuri 等人[47]则提出了用于 Microsoft SQL Server 的索引选择工具 AutoAdmin.

(1) 单查询优化. 对于单个查询的优化，往往都是通过对 Query 进行分析得到可索引列集合，然后将可索引列进行组合得到候选索引集合. AutoAdmin 会将工作负载 W 划分为只包含单个查询的工作负载 $W_1 \cdots W_n$. 对每一个工作负载 W_i，可索引列从 WHERE，GROUP BY，ORDER BY 和 UPDATE 子句中提取得到. 然后，使用一个迭代搜索算法 MC-ALL 进行配置枚举，迭代选择更复杂的索引集合，进而产生多列索引，以此类推. 每个索引方案的代价，依然采用该索引被使用后，查询执行时间的减少量来度量. 对于单个查询，DB2 Advisor 则使用 SAEFIS(Smart column Enumeration for Index Scans) 枚举算法枚举虚拟索引方案，通过对 SQL 语句分析，将可能建立索引的列分为五个集合，通过将这些集合进行组合，来产生虚拟索引方案集合，并统计相关信息.

(2) 工作负载优化. 使用单个查询得到的候选索引方案集作为起点，可以采用不同的方法组合得到工作负载的优化索引方案. AutoAdmin 将每一个负载 W_i 上进行配置枚举产生 C_i 进行集合并操作得到候选索引集合 C，先从候选集合中选出 m 个最优的索引方案作为种子. 然后，采用贪心的策略，每次将种子的大小扩大 1，即加入一个新的索引，使得每次加入索引获得的收益最大，直到种子扩充为 k 大小，即包含 k 个索引. DB2 Advisor 同样将索引选择问题模拟为背包问题处理. 但是，直接应用背包问题解法没有考虑负效益的积累，没有考虑索引之间的相互影响. 于是，DB2 Advisor 增加了 TRY_VARIATION 这一步骤改善背包问题的初始解. TRY_VARIATION 将解中的一小部分索引集合和不在解中的一小部分索引集合交换，然后重新使用这个变种解析工作负载，如果这个变种收益更多，那么就是当前解，否则仍保持原来的解，重复进行这一过程，直到优化时间预算耗尽.

AutoAdmins 给出的解决方法能够通过参数 k 控制索引方案中索引的数量，较为灵活. DB2 Advisor 最后的随机交换过程虽然可能解决简单应用背包问题的不足，但是随机过程不可控，不够稳定.

5.2.3　半自动化索引调整

由于在线和离线都存在一定的不足，于是，Schnaitter 等人[48]提出了一种称为半自动索引调整(Semi-Automatic Index Tuning)的技术，将在线方法和离线方法结合起来，充分利用二者各自的优点. 既减轻了 DBA 选择典型工作负载的工作量，充分利用了调优器的计算能力，同时，还将 DBA 的专业知识应用在内，实现迭代的索引调整方法. Schnaitter 等人认为候选索引方案选择以及一些其他组件可以从其他索引优化器中复用，将工作重点放在了如何利用 DBA 的反馈进行索引推荐.

随着候选索引方案数目的增加，为每一个候选索引子集合维护统计信息变得不可行. Schnaitter 等人提出了增强算法 WFA^+，采用分治的思想，对候选索引集合进行稳定划分，使子集之间不会相互影响，然后在每个子集上使用 WFA 算法求解，将每个子问题的解合并即可得到最终的索引推荐方案.

然而，WFA^+ 算法没有考虑反馈，为了避免这个问题，Schnaitter 等人提出了 WFIT 算法. 该算法将积极反馈和消极反馈分别保存在集合 F^+ 和 F^- 中，对于每一个候选索引子集 C_k 的推荐方案 $currRec_k$，新的推荐方案修改为 $currRec_k - F^- \cup (F^+ \cap C_k)$.

WFIT 算法以 WFA 为核心，存在一定的不足. 一方面，将工作负载视为序列，没有考虑一个索引方案对工作负载整体的影响. 另一方面，WFA 的性能严重依赖于候选索引枚举算法，无法在推荐中产生新的索引方案，这是必须考虑的一个问题.

5.2.4　基于机器学习的索引选择

Pedrozo 等人[49]提出了一种使用学习分类器与强化学习和基因算法相结合的索引调整方法 ITLS，这是目前已知的唯一将机器学习方法应用在索引选择问题上的研究. ITLS 使用学习分类器系统 LCS 创建和更新由规则表示的知识，并生成覆盖 DBMS 中绝大多数情况的调整索引来调整性能. 在典型的学习分类器系统中，基因算法将每个分类器视为一个个体，并将其中适应度较低的分类器替换为适应度较高的分类器. 每个分类器都基于监督学习，输入是索引选择，输出是性能的分值. 每个分类器对应一组推理规则表示，包括先行词和结果两部分，遵循“如果〈条件〉-那么〈动作〉”的格式. 每个个体(分类器)的染色体上有 16 个基因编码 16 个先行词(条件)，1 个基因编码结果(动作). 条件表示环境的一些特征，如表中是否存在索引等；动作表示对索引执行的操作，如创建、删除等. 每个分类器的强度用来表示在环境中的适应度，强度越高的越有可能被使用并且将其优势保持下去.

ITLCS的工作流程大概分为以下4步：

(1)检测器从环境中检测到信息(查询)，规则和信息子系统对信息进行处理，所有分类器尝试将信息与先行词组合；

(2)如果有多个满足条件的分类器，它们在环境中进行竞争，直到选出一个优胜者；

(3)获胜的分类器采取动作后，从环境中接收反馈信息，对获胜的分类器进行奖励或处罚，最后要对所有分类器征收生命税，降低其强度；

(4)使用遗传算法在分类器中进行进化，产生新的后代，淘汰不适合当前环境的分类器.

ITLS虽然将学习分类器系统应用在了求解索引选择问题中，但是它选择过程主要依赖遗传算法，并未将机器学习算法深入应用到问题的求解过程中.

5.2.5 未来工作

目前，深度学习、强化学习等模型都已用在了数据库调参、连接优化等问题中，并且取得了很大成功.在索引选择问题上，使用人工智能方法求解该问题是一个很有价值的研究方向，挑战包括：

(1)为训练神经网络获取高质量的数据集；

(2)对于动态变化的工作负载，使用机器学习方法进行在线索引选择，要求机器学习模型需要具备很强的适应能力；

(3)DBA工作经验是很有价值的知识，如何学习这些知识并进行应用也是一个需要解决的问题.

6 物化视图自动选择

随着数据规模的日益增大，数据库查询面临着批量SQL查询效率低下的问题.实验表明[50-52]批量SQL查询中存在重复的子查询，如何减少批量查询的重复计算在查询优化研究领域变得越来越重要.其中，选择子查询建立物化视图并复用是降低批量查询中重复计算的重要手段.一般而言，选择子查询建立物化视图主要分为两大步骤：识别等价子查询和选择子查询建立物化视图.其中，两个子查询被称为等价子查询当且仅当他们对应的关系代数表达式是等价的，这表明两个等价子查询可以互相利用彼此的查询结果，同时也为复用子查询物化视图的正确性提供了保证.针对批量的SQL查询语句，物化子查询视图要求我们首先识别出不同查询语句之间的等价子查询作为建立子查询物化视图的候选集，然后根据实际情况中出现的约束条件构建子查询的选择优化问题，并设计算法解决相应的优化问题，从而尽可能提高批量SQL任务的查询效率.因此下面我们分别从等价子查询识别和等价子查询选择两个方面对相关技术进行介绍(见表5).

表5 物化视图自动选择技术的相关工作

领域	方法		优点	不足	适用场景	表现	适应能力
等价子查询语句的识别	基于符号表示的等价性判断[51]		基于规则判断；易于实现	只考虑符号表示的等价性；缺乏完备性	完备性要求不高，等价逻辑复杂	低	高
	基于逻辑语义的等价性判断[52-53]		基于一阶谓词逻辑的等价性判断；完备性更高	一阶谓词等价判断证明器的效率低；可扩展性差	完备要求高，等价逻辑简单	高	低
物化视图的选择	基于DAG的启发式方法	基于AND/OR图的贪心算法[54]	效率高；可行性有理论证明	贪心选择的过程未考虑维护视图的成本	单机，负载单一	低	高
		基于MVPP图的遗传算法[55]	基于贪心算法的改进；可以更快地求得最优解	有可能产生不可行解；收敛速度不确定	单机，负载单一	中	低
		基于Data Cube Lattice图的算法[56]	可用于解决分布式数据仓库环境下的视图选择问题	只能针对包含group by从句的视图或者查询	分布式，场景单一	高	低
	基于ILP的优化算法[57-58]		将问题形式化为二分图点边标记问题；可扩展性强；效率高	不支持实时优化；没有考虑视图的动态维护	离线，负载模式固定	中	高
	基于策略的视图动态选择[58-59]		动态维护物化视图	可扩展性较差；鲁棒性差	在线，负载更新频繁	高	低

6.1 识别等价子查询

一个子查询本质上是一个可以被执行的查询语句，所以等价子查询的识别本质上就是判断两个查询语句在逻辑上是否等价，也就是判断两个查询语句的关系代数表达式是否等价.查询语句的关系代数表达式可以映射为一阶谓词逻辑表达式.具体来说，查询过程可以粗略看作扫描表中每条记录并选择满足查询语句过滤条件的所有记录作为结果的过程.扫描表的每条记录可以看作一阶谓词逻辑表达式的变量，输出就是判断变量带入表达式的判定结果.如果结果为True，则该记录被选中，否则该记录被过滤.因此，判断两个查询语句的关系代数表达式是否等价可以看成判断两个一阶谓词逻辑表达式是否等价.现在的研究已经证明了在有限时间内一阶

谓词逻辑的等价性判断是不可确定的[51]. 因此，从完备性的角度来说，现有的等价查询识别方法只能识别出特定约束条件下的查询语句(比如只针对join 查询、select 查询等). 主要分两类：一类是基于符号表示的等价性判断，另一类是基于逻辑语义的等价性判断.

6.1.1　基于符号表示的等价判断

早期的查询语句等价性判断只从 SQL 表达式出发，即判断 SQL 语句的符号是否满足等价性. 这些符号一般包括扫描表的名称，相关字段的名称以及谓词的符号表示. 之后又出现对 conjunctive 查询等价性判断[51]，conjunctive 查询特指具有如下形式的查询：SELECT columnlist FROM rellist WHERE equialitylist. 其中，columnlist 指的是被选择的属性，rellist 是表的集合，equialitylist 是选择条件. 他们把 conjunctive 查询转换成包含关键符号信息的集合，然后通过判断集合的等价性来证明相应的 conjunctive 查询是否等价. 另一种方法是将查询语句转换为二维表示的形式(用 tableaux 表示). 他们将包含 select、project 以及 join 操作的查询语句转换成 tableaux，这样的表示类似于 conjunctive 查询，然后通过多项式的算法去解决等价性的判断. 为了把高代价的查询转化为低代价的查询，他们设计了查询等价转换模型，进一步提出了利用包含多个 tableaux 的集合来表示更加复杂的查询语句. 综上所述，早期的等价性判断方法主要从查询语句的符号表示入手，然后通过符号等价规则完成查询语句的等价性判断.

6.1.2　基于逻辑语义的等价判断

早期的方法不能从根本上解决查询语句逻辑等价的判定问题. 比如，对于两个条件语句“where $a!=0$”和“where $a>0$ or $a<0$”，基于符号表示的方法就无法判断谓词“!=”，“>”和“<”的逻辑关系. 为了解决这样的问题，Shumo 等人[53]实现了一个 SQL 查询等价证明器 Cosette，这个证明器可以支持 outer join 查询，aggreagte 查询，union 查询等多种查询的等价性判断. 具体来说，Cosette 将待判定的查询解析成符号关系上的约束(即一阶谓词逻辑表达式)，然后检查待判定的查询逻辑表达式是否存在不同返回结果. 如果存在，那么表明出现了等价性的反例，从而证明待判定的查询是不等价的[52]. Cosette 还利用基于 Homotopy Type Theory 设计的数学函数将查询编译成 K-relations 的表达式，并称其为 UniNomials，然后可以使用 Coq 证明器判定 K-relations 表达式的等价性[57].

6.2　自动选择子查询构建视图

给定批量的 SQL 查询语句，选择子查询建立物化视图的过程可以看做基于给定约束条件下(比如查询结果的基础要求和磁盘存储空间的有限性)的优化过程. 因此，选择子查询的问题可以形式化地定义如下：给定数据库表的集合 $R=\{R_1,R_2,\cdots,R_T\}$，以及批量查询负载 $Q=\{Q_1,Q_2,\cdots,Q_q\}$，优化目标是在给定约束条件下选择最优的视图集合 $M=\{V_1,V_2,\cdots,V_m\}$ 使得查询负载 Q 消耗最少的查询代价. 解决这个问题的传统方法主要分为两步：首先构建多个查询和子查询视图间的 DAG(Directed Acyclic Graph)，然后通过启发式的算法从构建的 DAG 图中选择最优的节点集合构建物化视图[54-56]. 随着分布式查询的广泛应用，一些研究者又提出了基于分布式环境的启发式算法用于解决海量查询负载下的视图选择问题. 随着数据量的增大，DAG 图也会随之增大，传统的启发式算法(如贪心算法、遗传算法等)普遍存在迭代求解时间长并且无法在有限时间得到最优解的问题. 为了解决这样的问题，研究人员提出了基于 ILP(Integer Linear Programming)问题的并行优化算法[57-58]. 此外，随着机器学习的发展，基于深度学习和强化学习的物化视图选择也成为研究的热点[58-59](见表 5). 下面我们分别介绍这三类视图构建技术.

6.2.1　基于 DAG 图的启发式方法

根据 DAG 图的种类，基于 DAG 图的传统算法可以分为三类，分别是基于 AND/OR 图[54]的贪心算法，基于 MVPP(Multi-View Processing Plan)图[55]的遗传算法和基于数据立方体格图[56]的算法.

AND/OR 图是由查询语句解析后的执行计划构成的，主要包含两种节点：操作节点和等价节点. 每个操作节点表示一个 SPJ(Select-Project-Join)代数表达式，而每个等价节点表示等价的逻辑表达式集合，即等价节点中的所有逻辑表达式对应的查询语句对应了同样的执行结果. 其中，操作节点连接的孩子节点只有等价节点，等价节点连接的孩子节点只有操作节点，他们在图中的连接方式是分层连接的. 最上层的根节点对应最终的查询结果，而最下层的叶节点表示了所有要扫描的表. 如图 2 所示，圆形代表等价节点，矩形代表操作节点，除根节点之外的操作节点都是由视图构成. 视图 V_1 对应了查询 Q_1，

根据连接关系可以知道 V_1 可以通过 V_4 和 R_1 或者 V_4 和 V_3 计算得到. 基于上述 DAG 图, Mistry 等人[54]证明了使用贪心算法在多查询中选择物化视图最优解是可行的. 他们提出的贪心算法就是迭代地从 ANG/OR 图中选择能够使整体查询代价最小的视图集合进行物化. 但是, 这种贪心选择没有考虑到维护视图的成本, 所以他们继续扩展相关工作并考虑如何优化视图的维护代价[54]. 具体来说, 他们的算法探索了不同视图之间的关系, 然后为视图构建高效的执行计划从而降低构建物化视图所需的代价.

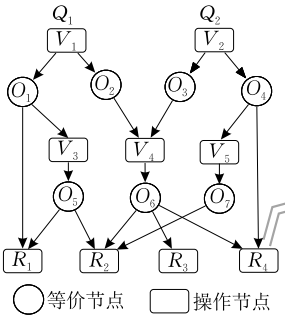

图 2 AND/OR 图示例

MVPP 图的表示和 AND/OR 图的表示十分相似, 其根节点代表不同的查询语句, 叶子节点代表基础的数据表. 不同之处在于 MVPP 图的中间节点都是操作节点, 而没有等价节点. 如图 3 所示, Q_1 的结果由操作节点 V_1 代表的视图执行后得到, 而 V_1 的结果又是由 V_4 和 V_5 代表的视图执行后得到. 基于 MVPP 图的算法[60]主要使用了遗传算法选择物化视图, 其中遗传算法寻找最优解的过程类似于自然界生物的进化. 遗传算法首先初始化一个随机解, 然后通过交叉和变异规则产生新的解, 直到在一段时间内最优解没有进一步的改善为止. 对于物化视图选择来说, 初始化的方法是根据物化视图的维护成本随机选择部分视图, 然后根据 MVPP 图中视图间的连接关系不断变换选择的视图获取反馈以更新可行解. 由于遗传算法的随机特性, 随机产生的一些解可能是不可行解. 为了避免不可行解的产生, 可以设计惩罚值作为优化目标的一部分, 加快遗传算法的收敛速度.

Ye 等人[56]提出了数据立方体格结构(Data Cube Lattice)为多维数据建模. Data Cube Lattice 中的每个节点对应为一个以 group by 从句的属性为特征的查询(或视图), 每条边表示了连接的两个节点的可导关系(derivation relationship). 如图 4 所示, 如果存在一条从 V_i 到 V_j 的路径, 那么 V_j 中的属性可以用 V_i 中的属性计算得到, 标记为 none 的节点对应了属性空集. 这种图表示的好处在于任意一个查询结果的属性可以被用于更新或者回答后继结点中查询结果的属性. Data Cube Lattice 在一些文献中[57]被用于解决分布式数据仓库环境下的视图选择问题. 分布式环境下需要解决视图和查询负载过大的情形, 通过扩展 Data Cube Lattice 图的概念可以自然地适应分布式环境. 具体而言, cube 中的所有节点对应到分布式环境所有计算节点上的查询视图, 如果不同计算节点间的视图存在可导关系那么就在相应的节点间增加连接边. 此外, Ye 等人[56]实现了分布式环境下的贪心算法, 他们首先选择一个初始的视图物化在存储空间最大的机器上, 然后贪心地选择其他和已选择的视图无可导关系的视图进行物化.

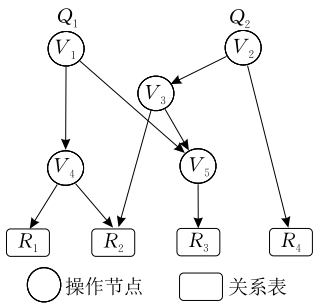

图 3 MVPP 图示例

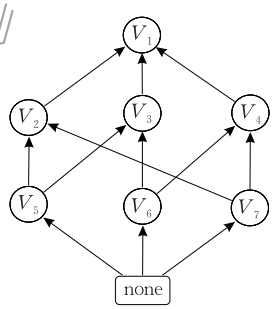

图 4 Data Cube Lattice 图示例

6.2.2 基于 ILP 的优化算法

给定查询负载和负载中每个查询对应的子查询, 我们选择子查询建立物化视图的优化目标是使得查询负载消耗的查询代价最低, Jindal 等人[57]将其形式化为 ILP 问题, 也就是在约束条件下子查询的 0-1 选择问题. 但是, 现有的 ILP 求解器无法在有限时间内解决大规模的子查询视图选择问题, 所以他们进一步将 ILP 问题简化为二分图上的点边的 0-1 标记问题, 然后使用基于概率的方法求得近似最优解. 具体来说, 二分图中的两类节点分别代表查

询和子查询，如果查询节点 Q 可以复用子查询节点 S 的结果，那么在这两个节点之间连接一条边，ILP 优化问题就演变为子查询节点和查询-子查询边的选择标记问题. 他们提出了迭代标记的算法，即首先随机标记一批子查询节点为 1，然后根据子查询节点的标记信息更新查询-子查询边的标记信息使得优化目标取得最大值，这样反复迭代直到优化目标收敛为止. 其中选择子查询节点的依据是为每个子查询节点计算被更改标记的概率. 为了解决大规模图计算的效率问题，他们提出了并行算法，即批量地计算子查询节点的概率并改变标记，加快了优化问题的收敛速度.

考虑到基于概率标记子查询的方法不能快速得到收敛的近似最优解，Liang 等人[58]提出利用强化学习技术提高二分图点边 0-1 标记问题的求解效率. 具体来说，他们将上述的迭代求解过程建模为 MDP(Markov Decision Process)过程，并且把当前子查询节点以及查询-子查询边的标记作为状态，然后通过 Q-learning 网络学习最优决策，这里的策略指的是根据当前二分图的标记状态决定下一步该如何标记节点. 此外，为了解决优化目标中涉及的子查询代价预估问题，他们利用了深度学习技术充分挖掘大量历史查询数据的特征并构建相关模型提高了子查询代价预估的准确性.

6.2.3　基于策略的物化视图动态选择

上述的物化视图选择方法旨在寻找最优的子查询集合建立物化视图，却没有考虑如何动态维护物化视图. 直观来说，提前选择的物化视图可能在以后的查询中被复用的频率逐渐降低，这就会导致物化视图的效用随着时间的推移逐渐降低. 如何在有限资源下动态选择物化视图成为解决这一问题的重要手段. 其中，Perez 等人[59]提出了为视图建立打分机制，即被物化的视图会根据查询负载(多个查询向量的叠加)的变化不断更新分值. 当存储空间不足以存储新的物化视图时，该算法会贪心地从已存储的物化视图中选择当前记录的分值最低的一些视图进行消灭以释放存储空间，而分值的计算是通过自定义的视图代价评估模型(真实值)得到的. 具体来说，当执行查询时，他们会用当前查询的执行代价更新与当前查询相关的所有视图的分值. 此外，Liang 等人[58]将物化视图的创建和删除统一成物化视图的动态选择过程，并且基于强化学习技术学习视图创建和删除的策略，为物化视图的动态选择提供了新思路.

6.3　未来展望

随着深度学习和强化学习的发展，越来越多的智能优化技术被用于数据库领域. 海量的历史查询数据为基于学习的数据库智能优化技术提供了数据支持，尤其是在智能建立物化视图的过程中. 从有监督学习的角度出发，我们可以构建包含查询、数据库、视图的有标签数据，即输入包括查询、候选视图和数据库的状态，标签是每个候选视图是否被选中. 从无监督学习的角度出发，我们可以利用强化学习的方法交互式地动态产生物化视图选择的最优解. 具体来说，历史数据只能提供经验信息，只有结合当前查询的反馈才能更好地选择最优的物化视图. 首先，我们可以针对不同的数据形式使用不同的网络结构抽取特征信息，比如对于查询语句的序列特性使用 RNN(Recurrent Neural Network)模型，而对于数据库状态信息可以使用 MLP(Multilayer Perceptron)模型. 然后，我们可以通过构建 MDP(Markov Decision Process)将物化视图的选择问题建模成迭代优化的问题，然后利用现有的深度强化学习方法解决.

7　研究展望与未来趋势

随着越来越多的智能优化技术在数据库领域得到应用，机器学习与数据库的关系日益密切. 机器学习为数据库处理效率和服务质量的提升提供了革命性的新思路，而数据库为机器学习算法提供需要的数据和应用场景. 本节中，结合现有的工作，我们进一步对未来机器学习与数据库系统的融合方式给出展望.

7.1　自治数据库

2017 年 Oracle 推出第一款自治式云数据库. 云数据库概念为分布式架构和具有隔离性、高可用性的数据服务提供了便利的部署条件，已经形成了成熟的产品链. 然而，数据库自治的概念刚刚兴起，其核心是利用机器学习技术实现数据库自动化：提供性能微调服务，针对负载特性自动优化执行和运维流程，并在保持运行的同时自动升级软件、打补丁，实现自驱动(Self-Driving)；在不停机的条件下动态调整计算和存储资源，甚至根据用户使用习惯动态改变资源配置，实现自扩容(Self-Scaling)；自动保护宕机事件，在宕机发生时能迅速回溯找到恢复策略，

实现自愈(Self-Repairing).基于以上提及的策略,未来自治型数据库要进一步实现自动管理、自动调优和自动组装:

(1)自动管理.数据库自管理能够利用机器学习技术,实现数据库自动检测,挖掘软硬件隐患;数据库自动修复,对不同级别的系统故障选择配套的处理策略;并利用可视化工具,将数据库状态的评估结果反馈给用户;

(2)自动调优.数据库自调优技术利用机器学习算法,动态感知外界环境和数据库内部变化,实时调整系统参数,合理配置任务处理、日志统计、监控分析等各方面机制;合理配置资源,提高资源的利用率;更新软件,及时打补丁,修补安全漏洞;

(3)自动组装.数据库自组装技术基于"组件仓库"的概念,它利用机器学习模型,根据当前用户负载和环境状态,动态组织查询的处理流,为用户提供精准、高效的个性化服务.

7.2 OLAP 2.0:多元异构数据管理

随着现代信息和网络技术的飞速发展,数据的管理和分析要面对多样的数据源(如 Digital Car 中,要对摄像头、雷达、超声波等多种传感器的数据进行融合)、数据结构和数据维度,如关系型数据、时空数据、流数据、音视频数据等.利用这些多元异构数据的关联、交叉和融合,使大数据的价值最大化,是数据库研究领域的一大挑战.传统多元信息融合技术采用信息关联、信息集成、信息过滤等方法,对从多个来源获取的信息进行处理[60].然而由于多源数据往往有着多种类型和表示形式,而且数据之间多具有复杂的关联关系[61-62].比如,基于阶段的数据融合方法,在各个阶段没有考虑异构数据之间的相互作用,失去了异构数据之间互补性的优势.因而这类方法不能跨越异构数据之间的语义鸿沟,实现真正内在的数据融合.深度学习技术由于具有综合学习、分析大量无监督数据的能力,与多元异构数据融合技术结合起来,可以帮助更全面地挖掘出目标信息[62-63].然而,当前的深度学习技术多用在单数据源的学习和研究上[64],在多元异构数据上的应用还不够成熟.未来 OLAP2.0 时代,要求数据库通过结合机器学习技术,高效存储、管理多元异构数据,应对混合型的事务处理、事务分析等负载需求.首先,学习模型能够记忆各种类型数据与存储方式的映射关系,存储模型可以自动根据新来数据的特征向量(eigenvector)选择合适的存储类型.其次,基于神经网络的反向传播机制高效融合多元异构数据,结合混合型负载进行分析,提高数据存储、管理、查询的效率.

7.3 机器学习在云数据库上的应用

云数据库服务(DBaaS)在提供免部署、高性能、高可靠和扩展灵活等特性的同时,也对各类数据库技术有了更高的要求.未来我们可以通过结合机器学习技术,训练计算机模型有记忆的探索数据模式,提高解决方案的智能化水平,使 DBaaS 产品能够自主运行.

(1)无服务器化.无服务器化(serverless)架构把数据库打包成服务,允许物理配置弹性变化,如 AWS Lambda、Microsoft Azure Functions、Google Cloud Functions 等.云供应商承担了服务器设备管理和维护的工作,需要管理大量数据库和物理资源.如之前讨论的,数据库的物理状态在基数估计、查询计划、索引创建等方面都是很重要的考虑因素.然而对于传统数据库,其基于的物理环境,如 CPU、内存、磁盘等,都是相对稳定的,短时间内不会有快速变化.因此,如读/写记录、磁盘页等物理操作的代价往往是静态设置.而且为了保证用户查询效率,减小监督进程的开销,磁盘利用情况等系统统计视图的更新频率往往设得较低.在云数据库场景下,不能直接套用这些策略,否则面对物理环境的改变,数据库很难快速适应.云数据库需要应对用户在线扩展的需求,用新的策略统计、分析和评估当前的物理环境.利用机器学习技术,模型可以对不同的环境特征进行学习,针对物理环境动态的调整配置.

(2)负载压力.提供云数据库服务的服务器集群要面对很高的访问压力,主要体现在四个方面:① 查询处理压力;② 资源调度压力;③ 网络传输压力;④ 多级别租户的负载调度压力.云数据库要应对不同的用户负载、处理指数级增长的多源数据;并根据负载特性、用户优先级、网络情况合理分配资源池和存储资源.通过结合机器学习技术,将智能融入到整个应用和工作流程中:通过在线预测负载开销,提前进行动态的资源、负载调度,如一套数据库系统供多个用户分时复用等,实现更加精细、精准、经济的负载管理.

(3)隐私和数据安全.隐私安全问题关乎云数据库厂商的信誉和用户的体验.企业重要数据和个人隐私数据一旦丢失,将会导致巨大的损失.然而到目前为止,以数据库为目标的暴力破解攻击占到了

40%.一旦秘钥被成功破解，数据库安全便不再存在.造成这个问题的原因主要包括两点：① 云租户在雇佣软件厂商完成应用开发后，往往没有专业技术获悉数据库中有哪些运维时留下的账号；② 因为不同数据库的加密算法不同，云租户很难发现数据库中的弱口令，给防止隐私泄露带来了难度.因此，可以利用机器学习技术，智能检查弱口令，如自动学习口令特点，发掘口令特征与安全性强弱的关系；基于复杂神经网络模型对明文做密文转换，智能地根据数据重要程度选择加密强度.

除了结合机器学习技术解决云数据库上的问题，面对云数据库上百万级的实例和大量用户，如何高效地训练学习模型也是一个问题.其一，可以利用HyperTune等工具，自动调整学习模型的超参数；其二，要充分利用数据库中大量的日志数据，线上、线下同时训练模型；最后，仍然需要DBA对学习模型做出的决策进行监察和把控，及时纠正机器学习中可能的过收敛、欠拟合等问题.

7.4　智能运维服务

随着IT系统与各类业务之间的联系日益密切.不少企业如银行已经达到了IT即业务的程度.然而，更多的业务需求使得IT系统更加复杂，管理难度和成本都随之增加.如何高效运维成为IT部门乃至CIO必须面对的问题.在这样的背景下，2016年Gartner提出智能运维(AIOps)，通过结合AI和大数据技术，期望在面对各类报警信息时快速找到问题所在并给出解决方案.但是自Gartner提出这个概念到现在，虽然在一些互联网和电信企业已经有项目落地，但仍没有人提出完整的理论体系或实施指南.究其原因.其一，运维工作涉及过多琐碎任务，而且对不同的系统往往有不同的统计、分析和监控方法，很难给出普适性策略.其二，AI技术与运维技术的结合还在探索阶段.由于深度学习模型本身的不可解释性，配置适合特定工作的模型往往匮乏指导性原则，需要凭经验和大量时间探索.因而现在IT运维团队多渐进式的部署AI组件，测试不同模型掌握知识的能力.未来工作可能需要提出一套完整的AIOps框架，根据特定任务适配相应的学习模型.

7.5　智能计算资源调度

未来面对ZB级别的数据体量，传统CPU数据库从数据计算速度到数据访问速度都将面临着瓶颈.而GPU等新的处理器由于其在大规模并行计算方面的出色表现，已经在深度学习领域被广泛用于浮点矩阵计算.目前已经有多家公司(如Sqream、Zilliz等)在做GPU数据库.他们利用基于GPU的众核处理器，进行并行数据处理.

GPU数据库加速数据处理的同时，也为数据库与深度学习、强化学习模型的融合提供了更加统一的硬件环境，数据库查询、网络训练都能在相同计算架构下执行.在基于人工智能的数据库框架下，一方面可以统一调配GPU等计算资源，并行实现数据库数据处理和神经网络训练.另一方面，可以考虑如何高效混合CPU、FPGA、GPU等有不同特点的处理器资源，按照用户需求动态调度计算资源.

7.6　基于自然语言的数据库查询技术

自然语言查询在提高数据库服务体验和运行效率方面有着很大的现实意义.近年来自然语言处理技术(NLP)被广泛研究并取得了长足发展，尤其是通过与各种深度学习方法的结合，自然语言模型已经被人和机器更好地理解[65].而NLP技术在语言表示、词法句法分析、语义理解乃至知识图谱等相关方向上取得了优异成果，也使基于自然语言查询的数据库应用成为可能.比如CRM软件服务商Salesforce的人工智能团队[66]提出用强化学习模型从自然语言生成结构化查询语句.他们构造了一个典型的序列到序列模型，在其中使用强化学习的方法，提高从自然语言SQL翻译的学习效率.但是为了简化翻译的难度，目前他们限制了支持的单词.未来通过进一步结合迁移学习、众包等技术，这项技术很可能更快学习到更加复杂的翻译知识，帮助数据库进一步平民化.

7.7　基于学习的数据库组织模式

当前数据库领域的研究主要集中在针对数据库局部问题进行优化.然而从各个部件的选择、组织到宏观的数据库架构设计，也都是非常有意义的问题.传统数据库在这方面主要有三个突出问题.其一，基于固定规则组织数据流，对负载特点和用户习惯的适应能力非常弱；其二，不能对所管理的数据分布情况进行学习，不能针对有不同分布的数据分别管理；其三，现有分布式的数据库架构存在着诸多缺陷."share-everything"模式，如Numa(Non-uniform memory access)，CPU访问内存的速度与节点的距离有关，在访问非本地节点的内存时速度慢."share-nothing"模式，如GaussDB，容错能力差，一个DN上操作失败会导致整个查询执行失败；计算

与数据强绑定,默认数据量大的节点计算能力就要强,配置非常不灵活.因此 Kraska 等人[15]提出了一种完全基于学习的数据库 SageDB.它用基于机器学习的组件完全代替数据库系统的核心组件,例如索引结构、排序算法和计划选择等.从总体上 SageDB 构建了一个能对数据负载分布感知的模型,并自动地为数据库每个组件选择合适的算法和数据特征.尽管这套系统目前还停留在理论论证层面,还有很多突出的问题没有彻底解决,但是它为未来的数据库架构和组织管理方法提供了非常大胆的方向.当每个组件乃至于决策系统都具有自主学习和优化能力,数据库将能更好地为用户服务.

7.8 数据库系统支撑机器学习

在 SageDB[15]中,它利用机器学习提高数据库工作组件的性能和适应能力.此外,我们通过将机器学习技术以组件的形式打包,可以使数据库更好地支撑机器学习.首先,用数据库降低机器学习的使用门槛.通过扩展关系代数,提供统一的查询接口,数据库可以同时提供数据操作和机器学习服务,简化机器学习的使用.其次,用数据库更好地支持机器学习.因为机器学习对数据的正确性、多样性有很高的依赖,我们通过在数据库中植入 AI 底座,原生提供典型的机器学习算法库,一方面可以很大程度上降低数据处理、传输的开销,提高机器学习的使用效率;另一方面,利用数据库中 B+树、哈希等索引结构,帮助提升机器学习算法的训练和执行效率.

此外,数据库机器学习化,不代表完全否定过去的工作.数据库优化领域也有很多杰出的设计理念和思想,可以用于提升机器学习模型的训练效率和表现.比如,利用物化视图方式组织、缓存不同任务下网络的权重分布情况,在相似场景下便不需要重新训练,利用网络的泛化能力便能很快适应;利用索引组织、管理乃至于重建神经网络,在使用多个网络模型的情况下,不仅可以提高网络的检索效率,而且可以改变传统网络基于随机游走的探索方式,更高效、有针对性的探索最优子空间,提高训练效率.未来,数据库与机器学习技术通过互相补益、融合,仍有极大的空间来提高各方面的表现和工作效率.

8 总 结

这篇文章中,我们综述了数据库技术的研究情况以及与机器学习技术结合的前景.面对海量异构数据和云场景下多样的用户和应用需求,数据库技术面临着多方面的挑战.我们首先概述了数据库的关键组件中的具体问题和与机器学习技术结合的方式,并给出了机器学习技术在数据库场景下可能遇到的问题.然后分别从数据库调参、基数估计、计划选择、索引自动推荐和物化视图自动选择五个方面介绍现有方法是怎么解决问题的.数据库调参方面,我们概述了专家决策、概率模型、启发式算法三种传统调参方法,对比介绍了传统机器学习方法、深度强化学习方法的优势和不足,给出未来工作;基数估计方面,我们概述了直方图、数据画像、基于索引的采样方法,对比介绍了分类器、深度学习方法的优势和不足,给出未来工作;计划选择方面,我们概述了动态规划、左深树、zig-zag 等静态的计划选择技术,对比介绍了深度学习、强化学习等基于机器学习的计划选择技术的优势和不足,给出未来工作;索引自动推荐方面,索引生成工作概述了范围索引、哈希模型索引、基于学习的布隆过滤,索引推荐工作分别概述了离线、在线、半自动化和基于分类器、强化学习、遗传算法等的机器学习方法,并给出与机器学习技术结合的未来工作;物化视图方面,等价子查询语句识别工作概述了符号表示、逻辑语义方法,而物化视图选择工作概述了多种基于 DAG 的启发式算法、基于 ILP 的优化算法和基于策略的动态选择算法,并给出与机器学习技术结合的未来工作.最后,我们讨论了自治数据库自动管理、自动调优和自动组装的发展方向,对未来机器学习与数据库关键技术的融合方式给出进一步展望.

参 考 文 献

[1] Ali R, Shahin N, Zikria Y B, et al. Deep reinforcement learning paradigm for performance optimization of channel observation-based MAC protocols in dense WLANs. IEEE Access, 2019, 7: 3500-3511

[2] Lin R, Stanley M D, Ghassemi M M, Nemati S. A deep deterministic policy gradient approach to medication dosing and surveillance in the ICU//Proceedings of the Engineering in Medicine and Biology Conference. Berlin, Germany, 2018: 4927-4931

[3] Wu E. Crazy idea! Databases ⋈ reinforcement-learning research //Proceedings of the Biennial Conference on Innovative Data Systems Research. California, USA, 2019

[4] Zhu Y, Liu J, Guo M, et al. BestConfig: Tapping the performance potential of systems via automatic configuration

tuning//Proceedings of the ACM Symposium on Cloud Computing. California, USA, 2017: 338-350

[5] Dikaleh S G, Xiao D, Felix C, et al. Introduction to neural networks//Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research. Toronto, Canada, 2017: 299

[6] Gallinucci E, Golfarelli M. SparkTune: Tuning spark SQL through query cost modeling//Proceedings of the Conference of the International Conference on Extending Database Technology. Lisbon, Portugal, 2019: 546-549

[7] Trummer I, Moseley S, Maram D, et al. SkinnerDB: Regret-bounded query evaluation//Proceedings of the International Conference on Very Large Data Bases. Rio de Janeiro, Brazil, 2018, 11(7): 800-812

[8] Li G, Zhou X, Gao B, Li S. QTune: A query-aware database tuning system with deep reinforcement learning//Proceedings of the International Conference on Very Large Data Bases. Los Angeles, USA, 2019: 2118-2130

[9] Van Aken D, Pavlo A, Gordon G J, Zhang B. Automatic database management system tuning through large-scale machine learning//Proceedings of the ACM Special Interest Group on Management of Data. Chicago, USA, 2017: 1009-1024

[10] Zhang J, Liu Y, Zhou K, et al. An end-to-end automatic cloud database tuning system using deep reinforcement learning//Proceedings of the ACM Special Interest Group on Management of Data. Amsterdam, the Netherlands, 2019: 415-432

[11] Tzoumas K, Sellis T, Jensen C S. A reinforcement learning approach for adaptive query processing. DB Tech Reports, TR-22, 2008

[12] Gani A, Siddiqa A, Shamshirband S, et al. A survey on indexing techniques for big data: Taxonomy and performance evaluation. Knowledge and Information Systems, 2016, 46(2): 241-284

[13] Ma L, Van Aken D, Hefny A, et al. Gordon: Query-based workload forecasting for self-driving database management systems//Proceedings of the ACM Special Interest Group on Management of Data. Houston, USA, 2018: 631-645

[14] Zong S, Ritter A, Mueller G, Wright E. Analyzing the perceived severity of cybersecurity threats reported on social media. arXiv preprint arXiv:1902.10680, 2019

[15] Kraska T, Alizadeh M, Beutel A, et al. SageDB: A learned database system//Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research. Toronto, Canada, 2017

[16] Li G, Zhou X, Li S. XuanYuan: An AI-native database. IEEE Data(base) Engineering Bulletin, 2018, 41(4): 70-81

[17] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning//Proceedings of the Conference of the International Conference on Learning Representations. San Juan, Puerto Rico, 2016

[18] Wei Z, Ding Z, Hu J. Self-tuning performance of database systems based on fuzzy rules//Proceedings of the International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery. Kunming, China, 2014: 194-198

[19] Gounaris A, Torres J. A methodology for spark parameter tuning. Big Data Research, 2018, 11: 22-32

[20] Nguyen N, Khan M H, et al. Towards automatic tuning of apache spark configuration//Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). San Francisco, USA, 2018: 417-425

[21] Gu J, Li Y, Tang H, Wu Z. Auto-tuning spark configurations based on neural network//Proceedings of the IEEE International Conference on Communications. Kansas City, USA, 2018: 1-6

[22] Arulkumaran K, Deisenroth M P, Brundage M, et al. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine, 2017, 34(6): 26-38

[23] Leis V, Gubichev A, Mirchev A, et al. How good are query optimizers really? Proceedings of the VLDB Endowment, 2015, 9(3): 204-215

[24] Zhang J, Xiao Z, Yang X, et al. Winslett, M. Differentially private histogram publication. Proceedings of the VLDB Endowment, 2013, 22: 797-822

[25] Tschorsch F, Scheuermann B. An algorithm for privacy-preserving distributed user statistics. Computer Networks, 2013, 54: 2775-2787

[26] Harmouch H, Naumann F. Cardinality estimation: An experimental survey. Proceedings of the VLDB Endowment, 2017, 11(4): 499-512

[27] Mon N, Matias Y, Szegedy M. The space complexity of approximating the frequency moments. Journal of Computer and System Sciences, 1999, 58(1): 137-147

[28] Durand M, FlajoletP. Loglog Counting of large cardinalities (extended abstract)//Proceedings of the 11th Annual European Symposium. Budapest, Hungray, 2003: 605-617

[29] Papapetrou O, Siberski W, Nejdl W. Cardinality estimation and dynamic length adaptation for Bloom filters. Distributed and Parallel Databases, 2010, 28(2): 119-156

[30] Leis V, Radke B, Gubichev A, et al. Cardinality estimation done right: Index-based join sampling//Proceedings of the Conference of the Biennial Conference on Innovative Data Systems Research. California, USA, 2017

[31] Kipf A, Kipf T, Radke B, et al. Learned cardinalities: Estimating correlated joins with deep learning//Proceedings of the biennial Conference on Innovative Data Systems Research. California, USA, 2019

[32] Ortiz J, Balazinska M, Gehrke J, Keerthi S S. Learning state representations for query optimization with deep reinforcement learning. Workshop on Data Management for End-To-End Machine Learning, 2018, 4: 1-4

[33] Marcus R, Papaemmanouil O. Plan-structured deep neural network models for query performance prediction. arXiv preprint arXiv:1902.00132, 2019

[34] Marcus R, Negi P, Mao H, et al. Neo: A learned query optimizer. arXiv preprint arXiv:1904.03711, 2019

[35] Sun J, Li G. An end-to-end learning-based cost estimator// Proceedings of the International Conference on Very Large Data Bases. Tokyo, Japan, 2020

[36] Giroire F. Order statistics and estimating cardinalities of massive data sets. Discrete Applied Mathematics, 2009, 157(2): 406-427

[37] Momjian B. PostgreSQL: Introduction and Concepts. Boston, America: Addison-Wesley, 2001

[38] Waas B, Pellenkoft A. Join order selection (good enough is easy)//Proceedings of the British National Conference on Databases. Kunming, 2010: 51-67

[39] Lang H, Neumann T, Kemper A, et al. Performance-optimal filtering: Bloom overtakes cuckoo at high-throughput. PVLDB, 2019, 12(5): 502-515

[40] Krishnan S, Yang Z, Goldberg K, et al. Learning to optimize join queries with deep reinforcement learning. arXiv preprint arXiv, 2018, cs. DB: 1808.03196

[41] Marcus R, Papaemmanouil O. Deep reinforcement learning for join order enumeration//Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management. Houston, USA, 2018: 3

[42] Yu X, Li G, Chai C, Tang N. Reinforcement learning with tree-LSTM for join order selection//Proceedings of the IEEE International Conference on Data Engineering. Dallas, USA, 2020

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

[44] Mitzenmacher M. A model for learned bloom filters and related structures. arXiv preprint arXiv, 2018, cs. DS:1802.00884

[45] Luhring M, Sattler U K, Schmidt K, et al. Autonomous management of soft indexes//Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop. Ankara, Turkey, 2007: 450-458

[46] Elghandour I, Aboulnaga A, Zilio D C, et al. An xml index advisor for DB2//Proceedings of the ACM Special Interest Group on Management of Data. Houston, USA, 2018: 1267-1270

[47] Alekh J, Shi Q, Hiren P, et al. Computation reuse in analytics job service at Microsoft//Proceedings of the ACM Special Interest Group on Management of Data. Houston, USA, 2018: 191-203

[48] Schnaitter K, Polyzotis N. Semi-automatic index tuning: Keeping dbas in the loop. PVLDB, 2012, 5(5): 478-489

[49] Pedrozo G W, Nievola J C, Ribeiro C D. An adaptive approach for index tuning with learning classifier systems on hybrid storage environments//Proceedings of the International Conference on Hybrid Artificial Intelligence Systems. Leon, Spain, 2018: 716-729

[50] Shumo C, Konstantin W, Alvin C, Dan S. HoTTSQL: Proving query rewrites with univalent SQL semantics// Proceedings of the ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software. New York, USA, 2017: 510-524

[51] Surajit C, Moshe V Y. Optimization of real conjunctive queries. PODS, 1993: 59-70

[52] Shumo C, Brendan M, Jared R, et al. Axiomatic foundations and algorithms for deciding semantic equivalence of SQL queries. PVLDB, 2018, 11(11): 1482-1495

[53] Shumo C, Chenglong W, Konstantin W, et al. Cosette: An automated prover for SQL//Proceedings of the Conference of the Biennial Conference on Innovative Data Systems Research. California, USA, 2017

[54] Mistry H, Roy P, et al. Materialized view selection and maintenance using multi-query optimization//Proceedings of the ACM Special Interest Group on Management of Data. Indianapolis, USA, 2010: 1267-1270

[55] Horng J T, Chang Y J, Liu B J. Applying evolutionary algorithms to materialized view selection in a data warehouse. Soft Computing, 2003, 7(8): 574-581

[56] Ye W, Gu N, Yang G, et al. Extended derivation cube based view materialization selection in distributed data warehouse// Proceedings of the 6th International Conference on Advances in Web-Age Information Management. Guangzhou, 2005: 245-256

[57] Alekh J, Konstantinos K, Sriram R, et al. Selecting subexpressions to materialize at datacenter scale//Proceedings of the International Conference on Very Large Data Bases. Rio de Janeiro, Brazil, 2018, 11(7): 800-812

[58] Liang X, Elmore J A, Krishnan S. Opportunistic view materialization with deep reinforcement learning. arXiv preprint arXiv, 2019, cs. DB: 1903.01363

[59] Perez L L, Jermaine C M. History-aware query optimization with materialized intermediate views//Proceedings of the IEEE 30th International Conference on Data Engineering. Chicago, USA, 2014: 520-531

[60] Fernandez R C, Termite S M. A system for tunneling through heterogeneous data//Proceedings of the ACM Special Interest Group on Management of Data. Amsterdam, Netherlands, 2019, 7: 1-7: 8

[61] Zheng Y. Methodologies for cross-domain data fusion: An overview. IEEE Transactions on Big Data, 2015, 1(1): 16-34

[62] Gheisari M, Wang G, et al. A survey on deep learning in big data//Proceedings of the IEEE International Conference on Computational Science and Engineering. Guangzhou, 2017: 173-180

[63] Zhang Q, Yang L, Chen K, et al. A survey on deep learning for big data. Information Fusion, 2018, (42): 142-157

[64] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436

[65] Hu D. An introductory survey on attention mechanisms in NLP problems//Proceedings of the 2019 Intelligent Systems Conference. London, UK, 2019: 432-448

[66] Zhong V, Xiong C, Socher R. Seq2SQL: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv, 2017, cs. CL: 1709.00103

LI Guo-Liang, Ph. D., professor. His interests include database, big data, database and machine learning hybrid techniques.

ZHOU Xuan-He, Ph. D. candidate. His interests include database and machine learning hybrid techniques.

SUN Ji, Ph. D. candidate. His interests include string similarity matching, and database and machine learning hybrid techniques.

YU Xiang, Ph. D. candidate. His interests include database and machine learning hybrid techniques.

YUAN Hai-Tao, Ph. D. candidate. His interests include trajectory management, and database and machine learning hybrid techniques.

LIU Jia-Bin, M. S. candidate. His interests include database and machine learning hybrid techniques.

HAN Yue, Ph. D. candidate. His interests include database and machine learning hybrid techniques.

Background

In the era of big data, in the face of ever-expanding data volume, complex and diverse application scenarios, different types of users, and heterogeneous hardware changes, traditional database techniques are difficult to adapt to these new scenarios and new changes. Machine learning, due to its strong learning and adaptability, has gradually demonstrated great potential and broad application prospects in the field of databases.

On the basis of full investigation and analysis, we first summarize the changes and opportunities that machine learning brings to database. Then we summarize the research status of database and machine learning hybrid techniques, and introduce the changes and opportunities brought by deep learning and reinforcement learning to the database in the aspects of automatic parameter tuning, automatic cardinality estimation, automatic query plan selection, automatic index and view selection. Finally, we provide future research directions and challenges.

This work was partially supported by the National Natural Science Foundation of China (No. 61632016), the National Grand Fundamental Research 973 Program of China (2015CB358700), Huawei and TAL Education.