软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(2):817−859 [doi: 10.13328/j.cnki.jos.007506] [CSTR: 32375.14.jos.007506]

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

AI 赋能的关系型数据库系统研究: 标准化、技术与挑战*

姬 涛[1,2], 钟 锴[1,2], 李奕言[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)

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

摘 要: 随着大数据时代的到来, 海量数据应用呈现出规模性 (volume)、多样性 (variety)、高速性 (velocity) 和价值性 (value) 的典型特征. 这种数据范式对传统数据采集方法、管理策略及数据库处理能力提出了革命性挑战. 近年来, 人工智能技术的突破性发展, 特别是机器学习和深度学习在表征学习能力、计算效率提升及模型可解释性方面的显著进步, 为应对这些挑战提供了创新性解决方案. 在此背景下, 人工智能与数据库系统的深度融合催生了新一代智能数据库管理系统. 这类系统通过 AI 技术深度赋能实现了交互层、管理层、内核层这 3 大核心创新: 面向终端用户的自然语言交互; 支持自动化运维的数据库管理框架 (如参数调优、索引推荐、数据库诊断和负载管理等); 基于机器学习的高效可扩展内核组件 (如学习索引、智能分区、智能查询优化、智能查询调度等). 此外, 新兴的智能组件开发接口 (API) 进一步降低了 AI 与数据库系统的集成门槛. 系统性地探讨智能数据库的关键问题, 以“标准化”为核心视角, 提炼出各研究主题 (交互范式、管理架构和内核设计) 内在的通用处理范式和特征. 通过深入分析这些标准化的流程、组件接口与协作机制, 揭示驱动智能数据库自优化的核心逻辑, 综述当前研究进展, 并深入分析该领域面临的技术挑战与未来发展方向.

关键词: 数据库系统; 数据管理; 人工智能; 机器学习

中图法分类号: TP311

中文引用格式: 姬涛, 钟锴, 李奕言, 李翠平, 陈红. AI赋能的关系型数据库系统研究: 标准化、技术与挑战. 软件学报, 2026, 37(2): 817–859. http://www.jos.org.cn/1000-9825/7506.htm

英文引用格式: Ji T, Zhong K, Li YY, Li CP, Chen H. Empowering Relational Database Systems with AI: Standardization, Technologies, and Challenges. Ruan Jian Xue Bao/Journal of Software, 2026, 37(2): 817–859 (in Chinese). http://www.jos.org.cn/1000-9825/7506.htm

Empowering Relational Database Systems with AI: Standardization, Technologies, and Challenges

JI Tao[1,2], ZHONG Kai[1,2], LI Yi-Yan[1,2], LI Cui-Ping[1,2], CHEN Hong[1,2]

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

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

Abstract: The advent of the big data era has introduced massive data applications characterized by four defining attributes: volume, variety, velocity, and value. These attributes pose revolutionary challenges to conventional data acquisition methods, management strategies, and database processing capabilities. Recent breakthroughs in artificial intelligence (AI), particularly in machine learning and deep learning, have demonstrated remarkable advancements in representation learning, computational efficiency, and model interpretability, thus offering innovative solutions to these challenges. This convergence of AI and database systems has given rise to a new generation of intelligent database management systems, which integrate AI technologies across three core architectural layers: (1) natural language

* 基金项目: 国家重点研发计划 (2023YFB4503600); 国家自然科学基金 (U23A20299, U24B20144, 62172424, 62276270, 62322214)
收稿时间: 2025-04-21; 修改时间: 2025-06-03; 采用时间: 2025-07-23; jos 在线出版时间: 2025-09-02
CNKI 网络首发时间: 2025-11-06

interfaces for user interaction, (2) automated database administration frameworks (including parameter tuning, index recommendation, database diagnostics, and workload management), and (3) machine learning-based efficient and scalable components (such as learned indexes, adaptive partitioning, query optimization, and scheduling). Furthermore, new intelligent component application programming interfaces (APIs) have lowered the integration barrier between AI and database systems. This study systematically investigates intelligent databases through a standardization-centric framework, delineating common processing paradigms across the research themes of interaction paradigms, management architectures, and kernel design. By examining standardized processes, interfaces, and collaboration mechanisms, this study uncovers the core logic enabling database self-optimization, reviews current research advancements, and provides an in-depth analysis of the technical challenges and prospects for future development.

Key words: database system; data management; artificial intelligence (AI); machine learning

经过半个世纪的研究, 关系型数据库管理系统 (relational database management system, RDBMS) 已经建立了坚固的理论基础, 积累了丰富的实践经验, 并在众多领域实现了广泛应用. 数据库技术的诞生和发展彻底革新了计算机数据管理的范式. 随着大数据时代的到来以及大数据应用场景的日趋复杂, 数据库系统面临着许多新的挑战. 为了满足大数据应用的实时性和场景多样性需求, 各种数据库相继问世, 极大地扩展了数据库的研究领域. 同时, 以机器学习和深度学习为代表的人工智能技术取得了显著成果, 数据库领域也应用人工智能技术应对其面临的挑战. 智能数据库系统已经成为数据库研究领域的一个前沿热点. 人工智能技术推动 RDBMS 朝着更高级别的智能化和自动化方向发展.

AI 赋能的智能关系型数据库管理系统是指将机器学习、深度学习等人工智能技术深度集成到数据库系统中, 旨在实现数据库的方便易用、高效管理和自主优化. 智能数据库系统通过对数据库查询负载、硬件特征、数据分布、查询日志和外部信息等进行特征的抽取和学习, 为数据库用户和数据库管理员 (database administrator, DBA) 提供了更易用和智能化的功能支持. 此外, 智能数据库系统还通过人工智能技术对数据库内核组件进行优化, 从而提升数据库的运行效率.

如图 1 所示, 智能数据库的发展主要围绕以下主题: 1) 智能交互层. 智能数据库通过创新的交互方式, 显著提升了用户的查询和数据处理体验. 与传统数据库依赖复杂的 SQL 语言不同, 智能数据库允许用户使用自然语言进行数据访问和操作, 极大地降低了技术门槛. 2) 智能管理层. 智能数据库利用人工智能的方法, 以辅助 DBA 或者完全自动化的方式为数据库正常稳定运行提供数据库调优与诊断, 负载分析和管理等功能. 具体而言, 数据库调优与诊断涵盖参数调优、索引推荐和数据库诊断; 负载分析与管理包括负载预测、负载生成和负载检测. 3) 智能内核层. 数据库内核组件是数据库内部相对独立的功能模块 (如数据存取, 查询优化和查询执行等), 智能数据库通过人工智能的方法优化或者完整替换内核组件来提升这些内核组件的性能. 其中, 数据存取涉及学习索引与数据分区; 查询优化包括查询重写、规模估算、代价估算与计划优化; 查询执行则涵盖自适应查询处理、并发控制与查询调度. 智能数据库系统和 3 个主题之间通过标准化的智能组件开发接口协同工作, 智能组件开发接口也提供对各个主题的统一的管理方式.

智能数据库系统以自驱动为目的, 是一种随着数据库及其工作负载随时间的变化而自动配置、管理和优化自身的数据管理系统. 智能数据库的设计包含了以上 3 个主题, 使得数据库可以自主运行无需人为干预. 虽然这 3 个主题 (交互层、管理层、内核层) 追求的目标 (自驱动) 一致, 但其各自内部实现智能化的具体过程却存在着相互独立的标准化范式. 实现数据库“自驱动”能力的各种智能方法, 虽因作用域 (如查询交互、系统运维、执行优化) 不同而呈现具体形态的差异, 但其运作机理通常体现为一种内在的、领域特定的闭环处理范式.

本文贡献如下: 聚焦于智能数据库系统的最新研究进展, 以标准化为核心视角, 系统梳理了各研究主题下的核心方向及其内在特征. 通过深入分析, 揭示了驱动智能数据库自优化的标准智能处理逻辑, 为研究者提供了面向不同研究方向的标准化学术框架, 明确了各方向下的关键问题与技术实现路径. 同时, 探讨了智能数据库的未来发展趋势与研究挑战, 为后续研究提供了方向性指导. 相较于已有综述文献[1,2]对智能数据库领域的细分及其研究进展与趋势的探讨, 以标准化视角进行归纳总结, 进一步从智能数据库的 3 大研究主题出发, 基于最新研究成果, 并提出了未来研究挑战与发展方向.

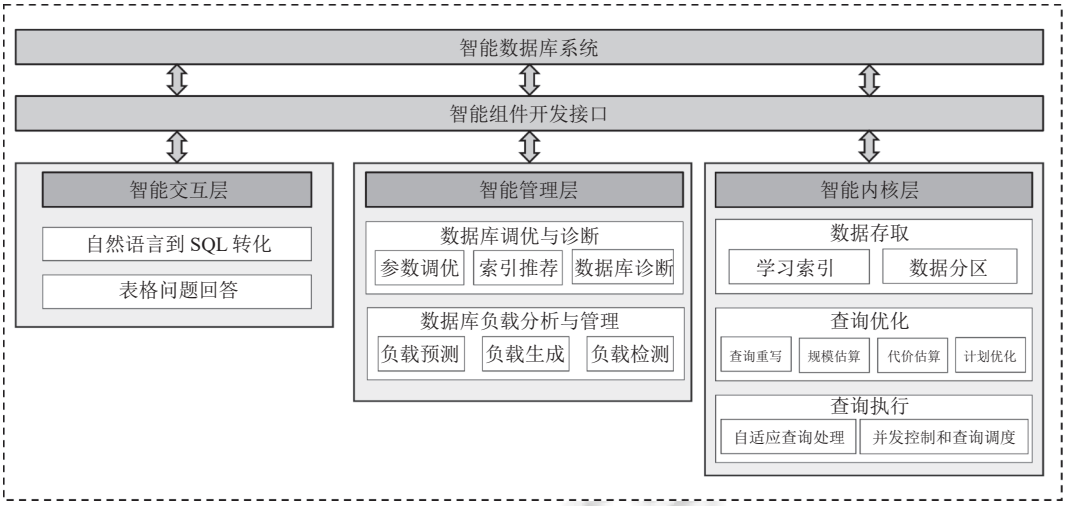

图 1 本文结构

1 智能数据库系统

智能数据库系统是通过集成人工智能技术实现自动化数据管理、优化查询性能并支持智能决策的下一代数据库解决方案. 智能数据库的设计采用了智能方法与数据库交互的标准形式. 智能方法通过智能组件开发接口与数据库进行通信. 智能数据库的可操作部分包括查询、配置和内核组件等. 这些部分之间可能存在相互影响, 共同决定数据库的性能. 例如, 索引推荐与查询优化可以协同工作, 通过动态调整索引策略和优化查询执行计划, 以实现数据库的最佳性能.

本节将介绍具有代表性的智能数据库研究包括 SageDB[3,4]、MB2[5]和 OpenGauss[6]. 除此之外, 本节还总结了其他商业数据库的智能化情况. 表 1 对这些智能数据库系统进行了横向对比并总结了这些智能数据库系统. 最后, 本节总结了智能数据库系统的未来发展趋势和仍然面临的挑战. AI4DB (AI for database) 和 DB4AI (database for AI) 是两个相互关联的技术方向, AI4DB 关注如何利用 AI 技术提升数据库的智能化水平, 而 DB4AI 则研究如何利用数据库技术优化 AI 模型的训练和推理过程[2]. 本文更多偏向于 AI4DB 的讨论, 一些数据库也会集成 DB4AI 的功能, 我们在本节进行了概述.

表 1 智能数据库系统

数据库	数据库类型	智能功能	智能化	完善度
SageDB	关系型数据库	学习索引、数据布局优化和计划优化	高	中
NoisePage	关系型数据库	自动索引优化、查询计划优化、硬件容量扩展和SQL调优	高	中
OpenGauss	关系型数据库	参数调优、索引推荐、慢SQL检测、查询重写、异常检测、计划优化等	高	高
NeurDB	关系型数据库	自主数据分析、自适应系统优化和动态数据管理	高	中
Oracle	关系型数据库	高级分析功能、自动化管理、实时应用测试和机器学习集成	中	高
Azure	云数据库	自动扩展、多模型支持和实时数据分析	中	高
Aurora	云数据库	故障检测与修复和自动扩展功能	中	高
Google Cloud	云数据库	机器学习集成、多区域复制和实时数据分析	中	高
SAP HANA	内存数据库	实时分析、数据压缩、高级预测分析和机器学习集成	中	高

1.1 SageDB

SageDB[3,4]整合不同的数据库智能组件, 以实例化优化的方式构建完整的数据库系统. 实例优化就是基于数据集和工作负载实例化数据库, 然后对其进行优化从而使得数据库接近专家解决方案的性能. SageDB 引入了两种实

例优化技术: 部分物化视图和复制数据布局. 部分物化视图是建立比传统的物化视图更加细化的物化视图方法, 复制数据布局与数据分区类似将实例优化数据布局的思想和表复制相结合. 同时, SageDB 还针对上述两种算法引入了一种针对数据库数据和工作负载特点的全局优化方法, 这种方法能够优化数据库全局而非单个数据库组件. SageDB 包含了以下 3 个设计原则: 1) 避免软件回归 (software regression). 新的组件避免对数据库产生任何负面影响. 2) 自动优化. SageDB 希望用户可以以最少的操作得到最优的性能. 3) 避免干扰. SageDB 系统地考虑数据库各个组件之间的干扰和影响, 避免了组件之间的负面影响. 除此之外, SageDB 还包含了许多智能组件, 比如学习索引、计划优化等.

1.2 NoisePage

ModelBot2 (MB2)[5]是 NoisePage 针对数据库管理系统难以部署和管理的问题, 提出的自治数据库管理系统方法. MB2 用人工智能的方法提升自治数据库管理系统的性能以及减少并发执行期间的干扰, 使得数据库管理系统的规划组件能够推断出新操作的预期影响. 与 SageDB 截然不同, MB2 的核心理念在于解构数据库管理系统 (DBMS) 的内核架构, 将其转化为独立的操作单元. 随后, 对这些独立的单元进行单独的建模和编程, 并以此来预测 DBMS 的当前状态和执行成本. 在线推理过程中, DBMS 结合这些组件模型预测系统状态和工作负载性能. 为了支持多线程和并发环境, MB2 将上述模型的输出定义为一组可测量的性能指标来估计操作组件之间的干扰. 除此之外, MB2 还设计了一种有利于训练自治数据库管理系统的数据生成和训练方法使得 MB2 训练操作组件的模型独立于工作负载和数据集.

1.3 OpenGauss

OpenGauss[6]构建了自治的数据库框架, 面向 DBA 提供了自动化的运维工具和面向数据库内核集成了多种智能数据库组件, 提升了数据库的自我管理能力. 其中, OpenGauss 面向 DBA 提供数据库管理和部署工具, 包括参数调优和诊断、索引推荐、慢查询检测和分析、负载趋势预测和数据库异常检测等工具. 面向数据库内核集成了查询重写、基数估计和计划优化. 除此之外, OpenGauss 为了方便部署和管理学习模型, 设计了有效的数据管理和模型训练平台. 同时, OpenGauss 还支持验证学习模型的有效性. 对此 OpenGauss 引入原生 AI 算子, 简化操作流程, 充分利用数据库优化器、执行器的优化与执行能力, 获得高性能的数据库内模型训练能力.

1.4 NeurDB

NeurDB[7]是一个由人工智能驱动的自主数据库系统, 旨在解决传统 DBMS 在面对动态数据和负载变化时的局限性. 首先, NeurDB 能够适应数据和负载的动态变化, 确保 DBMS 在动态环境中保持高可用. 为了应对数据和负载的变化, NeurDB 采用了增量更新的策略, 当检测到数据或者负载的变化时, 只对模型的部分进行微调, 以减少微调的成本使得模型快速适应变化. 微调后的模型会被记录到模型库中以根据需要快速切换到新的模型. 第二, NeurDB 提供了自主的系统优化和智能的数据库分析功能, 减少了用户的操作负担. 通过引入学习型的并发控制算法和查询优化器, NeurDB 能够自动调整系统行为以适应不断变化的数据和负载, 从而提高了系统的整体性能和可靠性. 最后, NeurDB 扩展了 SQL 语法, 引入了 PREDICT 关键字, 使用户能够轻松提交复杂的 AI 分析任务.

1.5 其他智能数据库

除了上述智能数据库系统外, 许多传统的商用数据库也集成了智能方法. Oracle[8]提出了智能数据库 Oracle autonomous database, 该数据库能够进行自动化数据库维护任务 (如自动升级和调优) 从而提升数据库效率和安全性. Azure database[9]集成自动调优功能, Azure 能够使用内置的机器学习模型自动调整索引来优化查询性能. Amazon Aurora[10]直接集成了机器学习功能, Aurora[11]提供了 SQL 接口来调用机器学习模型, 从而简化应用调用机器学习模型的过程. 与 Aurora 类似的 Google Cloud 提出了智能组件 BigQuery ML, BigQuery ML 提供了无需将数据移出 BigQuery 就可以直接在数据仓库中创建和运行机器学习模型的功能. SAP HANA[12]提供了一系列智能功能, 包括智能工作负载管理、即用型的人工智能服务以及智能代码开发等, 这些功能共同促进了数据库的高效运行和维护.

1.6 小 结

智能数据库提供了基于人工智能方法的自动化的维护、自我调优的查询处理器、高级数据分析和集成的机器学习等功能. 这些数据库在提升原本数据库性能的同时, 让 DBA 能够专注价值创造活动, 而减轻 DBA 的日常的管理和维护工作, 也向用户提供了智能的自然语言查询方式. 智能数据库为处理大规模、复杂的数据提供了支持, 同时也为数据驱动的应用提供了比传统数据库更高效和便捷的数据管理方式. 本文提出的标准化视角是理解智能数据库系统架构与技术演进的关键. 其核心在于发现智能数据库系统在交互层、管理层、内核层实现智能化的过程均遵循一种内在的、可抽象和复用的闭环处理范式. 这种标准化范式通常包含几个关键环节, 本文对各个研究领域实现智能化的范式进行了总结和抽象. 智能数据库毫无疑问会成为数据库未来的重要发展方向之一. 人工智能技术在数据库领域的应用将不仅局限于内核侧, 还可以用来提高运维效率, 同时数据库技术也将服务于 AI 应用, 提供支持. 智能数据库技术需要支持实时数据的捕获、处理和分析, 提供及时、准确的数据支持. 智能数据库还会朝着多样化的、融合、创新的趋势发展, 多种新型智能数据库产品将会出现, 如 NoSQL[13]、Graph 数据库[14]等, 满足不同应用场景需求. 智能数据库系统的发展在实现其智能交互层、智能管理层和智能内核层这 3 大功能维度的同时, 一方面各个主题面临自我发展的挑战 (第 2.3、3.3、4.4 节), 另一方面也面临着一系列跨层级整合与优化的关键挑战.

1) 高质量数据基础的挑战 (涉及所有层级): 无论是交互层语义理解、管理层决策优化还是内核层性能提升, 其底层 AI 模型的训练均高度依赖于大量高质量数据. 训练数据不足或质量低下会直接影响模型的准确性和泛化能力, 成为整个智能数据库体系效能的瓶颈.

2) 智能组件深度整合到系统的复杂性 (核心挑战: 管理层与内核层): 将人工智能方法有效整合到现有稳定、高效的数据库系统中, 尤其是在管理层 (自动化运维策略) 和内核层 (自适应执行引擎、内置机器学习模型), 面临着显著的技术兼容性问题 (如计算范式差异、资源调度冲突) 以及对现有数据库架构 (如存储、事务、日志机制) 的适配挑战.

3) 数据安全与隐私风险 (核心挑战: 管理层与交互层): 智能数据库运行过程中, AI 模型的训练和推理 (尤其在内核层嵌入模型或管理层用户交互数据利用) 可能引入新的数据泄露、隐私侵犯、推断攻击等安全风险, 需要在整个数据生命周期 (从管理层数据访问到内核层处理) 增强安全保障机制.

4) 多智能模块协同优化的难题 (核心挑战: 内核层与跨层协同): 智能数据库趋向于集成多种功能各异的 AI 方法 (如索引推荐、查询优化器、异常检测). 在内核层实现这些模块的高效协作, 避免决策冲突 (如不同优化目标冲突、资源争用), 并使其相互促进而非掣肘, 是目前需要研究的核心系统架构问题. 此外, 新智能方法与原有非智能核心组件 (如传统查询引擎、事务管理) 的和谐共存与优化同样面临挑战.

这些跨层级的挑战是制约智能数据库进一步成熟和落地的关键所在, 仍需系统性的解决方案, 并对下一代智能数据库系统的架构设计提出了更高要求. 下面的章节我们将分别围绕智能交互层、智能管理层和智能内核层介绍各个主题所使用的标准化方法, 涉及的具体技术, 以及面临的挑战.

2 智能交互层

为了提高用户体验, 智能数据库系统提供了两种主要的用户友好型数据交互查询方式: 自然语言到 SQL 转化和表格问题问答用于简化用户的查询过程.

在数据库交互方面, DBMS 通过引入自然语言处理 (NLP) 技术, 一方面实现了从自然语言到 SQL 的转换 (Text2SQL), 另一方面实现了不通过 SQL 语句直接回答用户关于数据库表格的问题 (TableQA). 这使得不具备编程背景的用户也能够以日常用语的形式提出问题, 而系统则负责将这些问题转化为精确的 SQL 查询语句, 从而执行相应的数据库操作. 这种方式不仅降低了使用门槛, 还提高了查询效率, 让任何人都能轻松访问和利用存储在数据库中的信息.

2.1 自然语言到 SQL 转换

图 2 展示了将用户提出的自然语言问题转换为可执行 SQL 命令 (Text2SQL) 的标准化过程. 自然语言与 SQL

转化遵从图 2 中所示的流程. 具体的, 系统将用户输入的自然语言问题或查询统一传递给自然语言处理 (NLP) 模型, 该模块负责理解查询的上下文、意图和结构. 与此同时, 数据库模式也被分析以理解数据结构和不同表及字段之间的关系. 自然语言到 SQL 转化的模型主要有两种: 一种是编码解码模型, 一种是大语言模型. 对于编码解码模型来说, 经过 NLP 处理的查询被输入到机器学习模型的编码器组件中, 编码器将自然语言查询转换为捕捉其语义含义的结构化表示. 随后, 编码后的表示被传递给解码器组件, 解码器根据编码信息生成相应的 SQL 查询, 并进行输出精炼, 确保生成的 SQL 查询语法正确且语义准确. 对于大语言模型来说, 大语言模型通过直接接收用户问题和数据库模式的描述, 将其转化为 SQL 查询. 最终, 模型精炼后的 SQL 查询作为系统的最终输出, 可以被数据库所执行. 此外, 还有一个反馈循环, 可以根据生成的 SQL 查询的准确性来微调模型. 我们在接下来的章节中逐一分解和比较各方法在标准化流程中的异同.

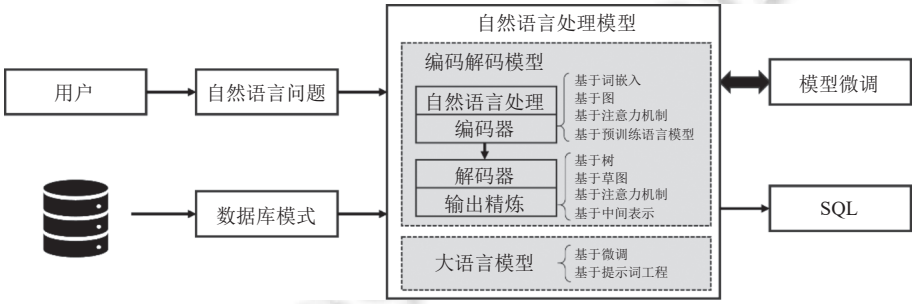

图 2 自然语言到 SQL 转换标准化架构

2.1.1 基于深度学习方法

(1) 编码. 编码就是将问题的语义表示, 数据库模式的结构表示, 数据库和查询的连接等问题, 利用深度学习模型统一编码成隐向量以包含丰富的语义和结构信息. 编码模型主要有以下 4 种, 分别是基于词嵌入的方法、基于图的方法、基于注意力机制的方法以及基于预训练语言模型的方法.

● 基于词嵌入的方法通过将查询中的每个单词映射到连续的向量空间, 使得语义相似的单词具有相近的向量表示. 该方法在 Text2SQL 的早期研究中得到广泛应用, 例如 Seq2SQL[15]、SQLNet[16]、IncSQL[17]和 TypeSQL[18]等模型均采用词嵌入方法对查询或数据库模式进行编码, 并将其作为输入. 与 one-hot 编码相比, 词嵌入方法能够为每个单词提供包含丰富语义信息的向量表示, 因为这些词向量可以从大规模语料库中学习到更优的词汇表征.

● 基于图的方法充分利用了数据库模式中蕴含的丰富结构信息. 这类方法通过图中的节点表示数据库中的表和列, 并通过边建模表与列之间的关系 (如包含关系、主外键约束等), 进而利用图神经网络 (GNN) 对图结构进行编码. 早期的代表性方法包括 GNN[19]、Global-GNN[20]和 RAT-SQL[21]等. 此外, 图结构还被用于编码自然语言查询, 从而与数据库模式编码相结合以解决 Text2SQL 问题. 具体而言, LGESQL[22]通过线图 (line graph) 捕获多跳语义信息, SADGA[23]则利用图结构为自然语言查询和数据库模式设计了统一的编码框架. S^2SQL[24]进一步关注问题中单词之间的语法依赖关系, 而 ShadowGNN[25]则对数据库模式进行抽象化处理, 忽略表或列的具体名称, 并通过图投影神经网络获取问题与数据库模式的联合表示. 这些方法显著提升了 Text2SQL 任务中对复杂语义和结构信息的建模能力.

● 基于注意力机制的方法主要采用基于 Transformer[26]的编码器, 通过注意力机制突出查询与数据库模式之间的关联关系, 从而优化编码表示并提升模型性能. X-SQL[27]、SQLova[28]和 UnifiedSKG[29]等模型直接利用 Transformer 作为核心模块, 充分发挥其强大的语义建模能力. 此外, RAT-SQL[21]和 DuoRAT[30]设计了关系感知的注意力机制, 专门用于捕捉表与列之间的复杂关系. 值得注意的是, 注意力机制是预训练语言模型 (如 BERT[31]、GPT[32]等) 的核心组件, 其在 Text2SQL 任务中得到了广泛应用. 这里特别总结了那些在模型中显著利用注意力机制的方法.

● 基于预训练语言模型的方法通过利用预训练语言模型 (如 BERT 等) 对查询问题和数据库模式进行编码, 以

解决 Text2SQL 任务中的编码问题. SQLova[28]和 RYANSQL[33]将输入问题中的单词与数据库模式中的单词串联后输入 BERT 编码器进行处理. 其他方法则利用预训练语言模型编码不同层次的信息, 例如, X-SQL[27]采用列编码替代段编码, 而 IRNet[34]等方法在此基础上引入了额外的特征以增强问题标记与表格列名之间的匹配. HydraNet[35]则使用 BERT 分别对问题和单个列进行编码, 以保持与 BERT 预训练任务的一致性. ETA[36]通过训练辅助概念预测模块来识别问题中与表和列相关的关键标记, 并通过检测删除问题标记后置信度得分的显著下降来确定重要标记. 此外, 一些方法提出了专门针对 Text2SQL 任务的预训练语言模型. 例如, TaBERT[37]利用表格数据进行预训练, 目标是通过屏蔽列预测和单元格值恢复任务来优化 BERT. GraPPa[38]则通过合成问题和 SQL 对预训练 BERT, 目标包括掩码语言建模、预测列是否出现在 SQL 查询中以及触发的 SQL 操作. GAP[39]在合成的文本到 SQL 和表格数据上进行预训练, 目标涵盖掩码语言建模、列预测、列恢复和 SQL 生成. 这些方法通过结合预训练语言模型的强大语义表示能力和任务特定的优化策略, 显著提升了 Text2SQL 任务的性能.

(2) 解码. 解码就是将编码器输出的查询和数据库模式的表示转化成 SQL 语句, 解码包括解码器解码和对解码结果的精炼输出. 解码模型主要有以下 4 个类别, 分别是基于树的方法、基于草图的方法、基于注意力机制的方法和基于中间表示的方法.

● 基于树的方法通过自上而下的解码器生成 SQL 的逻辑表达形式, 其中子树的组件基于其父节点和隐向量编码生成. Seq2AST[40]采用抽象语法树 (AST) 作为解码目标, 但该方法并未直接应用于 Text2SQL 任务. Syntax-SQLNet[41]则提出了一种针对 SQL 语法的基于树的解码方法, 通过递归调用模型预测不同的 SQL 组件, 直至生成完整的 SQL 查询. SmBoP[42]则提出了一种自下而上构建树的方法, 这种方法利用解码器对构造的解码树进行评分然后保留得分最高的几棵树. 基于树的方法充分利用了 SQL 查询的层次化结构特性, 能够更准确地捕捉 SQL 语句的语法和语义关系, 从而提升 Text2SQL 任务的生成效果.

● 基于草图的方法通过设计与 SQL 语法一致的草图框架, 并利用模型填充草图中的槽位来完成 SQL 查询的生成. 这种方法显著降低了 SQL 生成的复杂度, 因为模型只需预测草图中槽位的内容, 而无需学习完整的 SQL 语法和上下文信息, 但其性能也受到草图设计的限制. SQLNet[16]等方法是基于草图的典型代表, 它们设计了与 SQL 语法一致的草图框架, 并通过模型预测槽位内容. 文献 [43] 和 IRNet[34]进一步细化了基于草图的方法, 将解码过程分为两个阶段: 第 1 阶段粗略预测槽位内容, 第 2 阶段根据问题和第 1 阶段的预测结果填充更详细的草图内容. RYANSQL[33]则解决了利用草图生成复杂查询的难题, 通过递归设计多个草图框架, 并分别使用模型为每个草图填充槽位内容. 这些方法通过引入草图框架, 有效简化了 SQL 生成的复杂性, 同时提升了模型的可解释性和生成准确性.

● 基于注意力机制的方法利用 Transformer 模型整合解码端的信息, 通过计算注意力分数并与编码器的隐向量相乘, 生成上下文向量, 进而输出 SQL 字符. SQLNet[16]设计了基于注意力机制的解码器, 通过将列的隐藏状态与问题的嵌入相乘, 计算给定问题中列的注意力分数. 文献 [44] 在 SQL 组件选择中引入了对问题和列名称的双向注意力机制. 文献 [45] 则通过结构化注意力计算边际概率, 用于填充生成草图中的槽位. DuoRAT[30]在编码器和解码器中均采用了自注意力机制, 以增强模型对复杂关系的建模能力. 此外, 一些方法构建了序列到序列的 Text2SQL 系统, 直接使用标准的 Transformer 模型进行端到端的 SQL 生成. 这些方法通过引入注意力机制, 显著提升了模型对查询语义和数据库模式之间关系的捕捉能力, 从而提高了 SQL 生成的准确性和鲁棒性.

● 基于中间表示的方法通过设计 SQL 查询的中间表示形式, 使解码器首先生成中间表示, 再将其精炼为最终的 SQL 查询. IncSQL[17]定义了针对不同 SQL 组件的操作, 并让编码器和解码器对这些操作进行编解码. SmBoP[42]则利用关系代数作为 SQL 查询的中间表示形式. IRNet[34]引入了一种称为 SemQL 的中间表示, 通过解码器生成 SemQL, 再将其转换为最终的 SQL 查询. ValueNet[46]对 SemQL 进行了改进, 扩展了其对 SQL 语法的支持能力. 此外, NatSQL[47]和方法[48]等进一步改进和扩展了 SemQL, 使其能够更好地适应复杂的 SQL 生成任务. 这些方法通过引入中间表示, 降低了直接生成 SQL 的复杂度, 同时增强了模型对 SQL 语法和语义的建模能力, 从而提升了 Text2SQL 任务的性能.

表 2 总结了典型的基于深度学习的自然语言与 SQL 转化方法, 其中模型的输入涉及模型所接受的输入类型; 输出约束指系统为确保输出 SQL 语句的正确性而施加的约束条件; 模式链接指方法是否将数据库模式作为辅助信息纳入模型输入; 学习模型指 Text2SQL 模型在训练阶段采用的学习策略. 评测数据集指用于训练和评估 Text2SQL 模型的数据集. WikiSQL[15]和 Spider[49]提供了大规模多领域的 Text2SQL 数据集, 使得 Text2SQL 训练和测试标准化.

表 2 基于深度学习的自然语言与 SQL 转化方法

方法	模型输入	输出约束	模式链接	学习模型	评测数据集
Seq2SQL[15]	单词	序列	无	强化学习	WikiSQL
SQLNet[16]	单词	草图	无	—	WikiSQL
IncSQL[17]	单词	语法	无	—	WikiSQL
TypeSQL[18]	单词	草图	有	—	WikiSQL
SyntaxSQLNet[41]	单词	草图	无	—	Spider
GNN[19]	模式图	语法	无	—	Spider
Global-GNN[20]	模式图	语法/排序	无	—	Spider
SQLova[28]	序列	草图	无	迁移学习	WikiSQL
IRNet[34]	序列	语法	有	迁移学习	Spider
X-SQL[27]	序列	草图	无	迁移学习	WikiSQL
RAT-SQL[21]	模式图	语法	有	迁移学习	Spider
ValueNet[46]	序列	语法	有	迁移学习	Spider
HydraNet[35]	序列	草图	无	迁移学习	WikiSQL
SmBoP[42]	模式图	语法	有	迁移学习	Spider
RYANSQL[33]	序列	草图	无	迁移学习	Spider
ETA[36]	序列	序列	有	迁移学习	Spider

2.1.2 基于大语言模型方法

基于大语言模型的 Text2SQL 方法根据方法的实现方式, 分为基于提示词工程的方法和基于微调的方法.

基于提示词工程的方法通过设计特定的提示词 (prompt) 来引导大语言模型生成目标 SQL 查询. 这类方法无需对模型进行额外的训练, 而是利用大语言模型在预训练阶段学到的知识和推理能力, 通过自然语言提示词将任务描述转化为 SQL 查询. 基于提示词工程的方法依据其思考深度和对问题的分解能力可以分为以下 4 个类型.

● 朴素方法直接使用简单的提示词将自然语言问题与数据库模式结合, 输入到大语言模型中生成 SQL 查询. 一些研究[50,51]评估了零样本下的 LLM 在 Text2SQL 任务上的性能, 他们评估了不同的提示词和不同的 LLM 在 Text2SQL 任务上的性能. 结果表明 prompt 对于性能至关重要. 一些方法针对零样本下的方法进行改进, 提出了小样本的朴素方法, 他们提供了少量的样例来触发 LLM 以生成更准确的 SQL.

● 分解方法通过将复杂问题拆解为多个子问题, 逐步生成 SQL 查询. 一些方法[52−54]将 Text2SQL 任务分解, 先将问题分解为选择条件、聚合函数和表连接等子任务, 再分别生成对应的 SQL 片段, 最后组合成完整查询. 还有一些方法[55,56]将问题分解, 他们将用户的问题分解成不同的子问题, 然后通过解决这些子问题生成子 SQL, 再合成最终的 SQL.

● 推理增强方法[57−59]通过引入推理步骤或中间表示, 帮助模型更好地理解问题和生成 SQL 查询. 例如, 在提示词中加入逻辑推理链或中间结果, 引导模型逐步推导出正确的 SQL 语句. 这些方法通过思维链的提示词方法引导 LLM 一步一步地进行全面的推导过程.

● 执行优化方法[54,59,60]通过结合 SQL 执行结果对生成的查询进行反馈和修正. C3[61]将生成的 SQL 语句在数据库中执行, 并根据执行结果调整提示词或重新生成查询.

基于微调的方法通过任务特定的微调, 使大语言模型更好地适应 Text2SQL 任务. 这类方法通常利用标注的问题和 SQL 对数据集, 对预训练的大语言模型进行有监督训练, 以优化其在 SQL 生成任务上的表现. 例如, DAIL-SQL[62]设计为上下文学习框架, 采用采样策略提升少样本实例的效果. CodeS[63]则通过 ChatGPT[64]辅助的双向生成增强训练数据, 其预训练阶段分为 3 个增量阶段, 从基础的代码特定 LLM 开始, 逐步在包含 SQL 相关数据、自然语言到代码数据及自然语言相关数据的混合语料库上进行训练, 显著提升了文本到 SQL 的理解与生成性能. 此外, DTS-SQL[53]提出了一个两阶段分解的文本到 SQL 微调框架, 并设计了模式链接预生成任务, 以优化最终 SQL 的生成.

2.2 表格问题回答

表格问题回答 (table question answer, TableQA) 是另一种用户访问并向数据库发出查询的方法. 旨在从结构化表格数据中提取信息并回答用户提出的问题. 它结合了自然语言理解和表格数据的查询能力, 能够理解用户的问题, 定位表格中的相关数据, 并生成准确的答案.

如图 3 抽象出了 TableQA 的标准化流程, 首先, 用户通过自然语言提出问题. 系统随后将这些自然语言问题与数据库中的表格数据进行关联. 模型部分负责处理和理解这些数据, 进行问题消歧以确保准确理解用户意图. 接下来, 系统通过搜索检索机制在数据库中查找相关信息, 并通过迭代调优不断优化检索结果. 结果精炼阶段进一步处理和优化检索到的数据, 以提高答案的准确性和相关性. 最后, 模型微调步骤对系统进行精细调整, 以确保生成的答案符合用户需求.

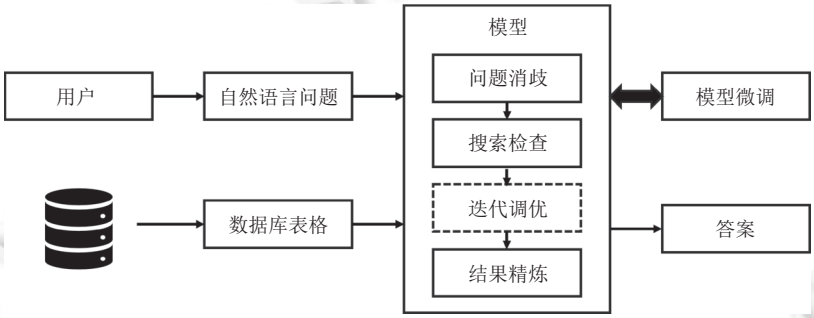

图 3 表格问题回答标准化架构

2.2.1 问题消歧

问题消歧是 TableQA 方法中的关键步骤, 旨在将用户输入的自然语言问题转化为一系列明确的中间操作或调整问题以消除歧义. TableGPT[65]引入了链式命令的概念, 通过微调大语言模型 (LLM) 将用户输入转换为一系列中间命令操作. 具体而言, TableGPT 利用大模型首先检查任务是否需要检索、数学推理、表格操作等, 并在问题过于模糊时提示用户无法回答. PACIFIC[66]将问题消歧分为两部分: 需求预测和问题生成. 需求预测用于判断是否需要进一步澄清问题的不确定性, 而问题生成则生成一个澄清问题以响应用户. 这些方法通过引入多轮对话和上下文理解, 显著提高了问题消歧的准确性.

2.2.2 搜索检索

搜索和检索是从结构化数据中提取相关信息以回答用户问题的核心过程. DocMath-Eval[67]发现, 检索模块能否发现最相关的结构化信息在很大程度上决定了 LLM 在 TableQA 中的准确性. 该方法使用现有的检索器从源文档中提取前 n 个最相关的文本和表格证据, 并将其作为输入上下文提供给模型以生成答案. Tap4LLM[68]探索了多种表采样方法 (如行和列采样) 和表打包技术 (基于令牌限制参数). 实验表明, 基于查询的采样方法能够检索与问题具有最高语义相似性的行, 显著优于随机采样、聚类采样等方法. cTBLS[69]设计了一个 3 步架构来检索并生成基于表格信息的对话响应. 首先, 基于双编码器的密集表检索模型标识与查询最相关的表. 接着, 利用基于三元组损失训练的系统状态跟踪模块, 对候选单元格进行相关性排序. 最后, GPT-3.5[70]生成以表格单元格为条件的自然

语言响应.

2.2.3 迭代调优

迭代调优是 TableQA 系统中优化模型输出的重要策略. 许多研究设计了迭代调用 LLM 的流程以提高结果的准确性和鲁棒性. GPT4Table[71]将复杂的任务分解为可管理的子任务, 通过多轮迭代逐步优化结果. SParC[72]根据新的用户输入动态更新模型输出, 利用多轮提示解决特定问题限制或纠正错误. 此外, 一些方法通过输出 Python 或 SQL 作为中间结果, 并将运行结果返回给模型进行进一步优化. 例如, DIN-SQL[54]通过多轮迭代生成和优化 SQL 查询, 显著提高了复杂查询的准确性.

2.2.4 结果精炼

与 Text2SQL 类似, TableQA 系统同样需要根据约束条件对查询结果进行精炼, 以满足用户对数字、类别、Python 代码或可视化结果的需求. 为实现这一目标, 许多方法采用了多样化的精炼策略. 例如, 当与 Power BI 等工具集成时[73], 模型的输出不仅限于文本, 还可以生成图表或其他可视化内容. TableGPT[65]便是一个典型示例, 它能够从文本和表格输入中直接创建可视化结果, 这一功能在数据分析任务中得到了广泛应用.

2.3 小 结

近年来, 智能数据库交互层的技术在自然语言处理与数据库系统的交叉领域取得了显著进展. Text2SQL 技术通过将自然语言查询自动转换为结构化 SQL 语句, 极大地降低了非专业用户访问数据库的门槛. TableQA 技术进一步扩展了数据库交互方式, 不仅支持 SQL 生成, 还能实现直接问答、多轮对话和复杂推理功能.

一些工作对 Text2SQL 进行了比较, 比如文献 [74] 聚焦于深度学习的 Text2SQL 技术发展, 系统梳理了基于深度学习的模型如何解决语义解析、模式编码和 SQL 语法生成这 3 大关键问题. 文献 [75] 着重梳理了 Text2SQL 的演进过程, 提供了 Text2SQL 技术演进的全景视图, 特别提出了大模型时代该任务仍然存在的挑战. 文献 [76] 则完全聚焦于大模型在 Text2SQL 领域的最新进展. 文献 [71] 探讨大语言模型对结构化表格数据的理解能力, 通过基准的设计评估 LLM 在表格数据上的表现. 文献 [77] 探讨了大模型解决结构化数据方面的应用, 梳理了 TableQA 任务的关键技术.

本节探讨的智能交互层技术发展体现了智能数据库标准化处理范式的应用. Text2SQL 流程从用户自然语言问题输入开始, 经过查询与模式的编码, 应用编码器-解码器模型或大语言模型进行 SQL 生成, 最终输出精炼的 SQL 语句并执行, 部分方法还利用执行结果反馈修正模型. TableQA 同样遵循这一范式, 在问题消歧、搜索检索、模型推理和结果精炼等步骤中体现了标准化流程的核心环节.

Text2SQL 和 TableQA 技术的未来发展前景广阔, 随着自然语言处理和大型语言模型的不断进步, 智能数据库系统将变得更加智能, 提供更准确、更高效的自然语言交互服务. 这些技术将扩展其跨领域和跨语言的能力, 支持更多领域的查询需求, 以及更多语言的自然语言输入, 实现更广泛的应用场景. 同时系统将更加健壮, 能够妥善处理模糊查询和复杂用户意图, 并提供有效的错误反馈和修正建议. 这些自然语言交互技术将融入各种数据相关工具和平台中, 进一步提升数据的可访问性和易用性.

Text2SQL 技术正快速迭代和发展, 并且吸引了广泛的研究兴趣, 但该领域依然面临两大尚未克服的挑战, 分别是自然语言的挑战和 SQL 带来的挑战. 1) 自然语言带来的挑战主要包含以下几方面, 首先是自然语言本身具有歧义的性质, 自然语言的歧义包括了词汇歧义、句法歧义、语义歧义、上下文歧义、省略推理等. 第二, 自然语言具有许多隐性知识, 某些查询需要模型不具有的隐式领域知识或常识, 从而导致查询生成的不正确或不完整. 第三, 自然语言语种具有多样性, 不同的人可能使用不同的语种进行查询. 2) SQL 带来的挑战, 首先 SQL 语句与自然语言不同, SQL 语句有严格的语法限制. 第二, SQL 语句与数据库的数据模式相关, 如何生成与当前数据库模式匹配的 SQL 语句带来了挑战. 第三, 词汇差距, 数据库词汇往往和用户使用的词汇在语义等方面不同, 预训练的语言模型可能无法表示这之间的词汇差距. 第四, 复杂查询, 现有的方法生成复杂的查询, 如嵌套查询、联接查询准确率较低甚至无法支持. 除此之外, 如何构建涵盖广泛查询类型和语言变化的合成训练数据集从而提高 Text2SQL 模型的鲁棒性和 Text2SQL 模型处理各种自然语言查询的能力也是一大挑战.

TableQA 技术在数据库表格问答领域仍面临若干关键性挑战以待解决. 首先, 在数值表示方面, 由于数值型数据在数据库表格中占据重要地位, 如何构建专门的数值表征方法成为提升问答系统性能的关键因素. 其次, 在复杂推理能力方面, 现有研究虽然已针对数据库表格开发了基于语义解析的推理方法, 但当前大多数非数据库表格问答系统仍局限于简单推理任务. 值得注意的是, 尽管大规模预训练模型展现出强大的潜力, 但其在数据库表格问答中的应用仍存在显著局限: 一方面, 模型的推理效率与结果准确性仍需进一步提升; 另一方面, 如何有效处理结构化数据以实现更精准的问答仍然是一个值得深入研究的核心问题.

3 智能管理层

智能数据库提供了新的数据库管理与维护方法, 旨在利用人工智能技术解决数据库管理员 (DBA) 在数据库运维中面临的挑战, 包括优化资源配置以提高数据库吞吐量和运行效率, 以及实现数据库的测试、监测和维护. 其核心分为两部分: 一是智能数据库调优与诊断, 涵盖数据库参数调优、索引推荐和数据库诊断, 通过智能算法优化配置参数、推荐索引策略并识别潜在问题; 二是智能数据库查询负载分析与管理, 提供负载预测、负载生成和负载检测功能, 分别用于预测未来负载趋势、模拟真实场景负载以测试性能, 以及实时监测负载并识别异常行为.

3.1 数据库调优与诊断

智能数据库调优与诊断是利用人工智能的方法对数据库的参数进行配置, 对索引进行推荐以及对数据库出现的异常进行诊断, 从而提升数据库效率的方法. 这些基于人工智能的方法不仅提升了数据库的运行效率和资源的利用率, 还降低了数据库维护和调试需要的人力资源和时间成本.

3.1.1 参数调优

数据库系统中存在大量参数配置, 这些参数在内存分配、数据读取、备份与恢复等方面对数据库性能产生显著影响. 智能数据库配置调优通过利用数据库负载信息、参数配置及系统状态等作为输入, 结合人工智能技术, 推荐一组最优参数配置.

图 4 是数据库参数调优的标准化流程, 数据库参数调优方法首先从数据库中获取参数、负载及系统状态数据, 随后通过预处理步骤筛选数据, 包括选择相关参数子集并对参数、负载及系统状态等特征进行提取. 接着, 编码后的特征被输入至调优模型中进行训练. 调优模型主要分为 5 类: 基于启发式算法、基于贝叶斯优化、基于深度学习、基于强化学习的调优方法以及基于大模型的方法. 最终, 调优模型输出优化后的参数配置. 此外, 为适应数据库运行状态及负载的动态变化, 调优模型需进行迁移, 其过程包括负载匹配、构建历史数据库及模型调整这 3 个步骤. 表 3 总结了智能参数调优方法及其主要特点.

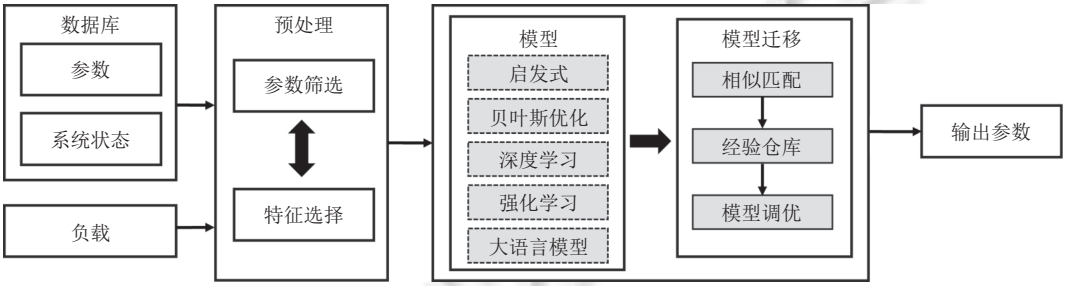

图 4 数据库参数调优标准化架构

表 3 智能参数调优方法

类别	调优方法	特征空间	调优模型	模型迁移	特点
基于启发式搜索的调优	Chen等人[78]	无	进化搜索	经验转移	发掘参数之间的依赖关系, 重组新旧样本
	BestConfig[79]	无	分割和发散采样, 递归定界与搜索	无	将参数子空间离散化

表 3 智能参数调优方法 (续)

类别	调优方法	特征空间	调优模型	模型迁移	特点
基于贝叶斯优化的调优	iTuned[80]	进化算法	响应曲面的高斯过程表示	无	提出了采用贝叶斯优化方法进行参数调优
	OtterTune[81]	线性回归特征选择技术	高斯过程回归	工作负载映射	利用调优历史数据, 重用历史观察结果
	Tuneful[82]	Gini重要性(杂质重要性)	贝叶斯模型+代价模型	工作负载映射	提出了一种调优成本摊销模型; 支持分布式数据库Spark
	ResTune[83]	DBA设定	单目标约束贝叶斯模型	超参数优化的元学习	预训练的调优模型提高迁移能力
	CGPTuner[84]	无	适应性上下文贝叶斯优化	工作负载映射	适应工作负载变化
	OnlineTune[85]	DBA设定, 安全评估	复合内核的贝叶斯优化	环境映射	在线动态调优, 关注调优的安全性
	LlamaTune[86]	低维空间投影, DBA设定	基于顺序模型的算法配置SMAC/高斯过程/DDPG	无	多模型集成
	KeenTune[87]	参数可解释性算法, DBA设定	随机搜索、基于高斯过程的贝叶斯优化、树状结构估计器潜在作用蒙特卡洛树搜索	工作负载映射	集成SHAP方法用于参数选择
基于深度学习的调优	IBTune[88]	负载信息和指定数据库指标	深度神经网络	无	利用缓存未命中率和缓冲池大小之间的关系优化云数据库的内存分配
	LITE[89]	负载信息和指定数据库指标	通过代码和调度的神经估计器	基于对抗学习的自适应模型	运行过程中对模型反馈, 小样本预训练提高了模型训练的效率
基于强化学习的调优	CDBTune[90]	DBA设定	DDPG	模型微调	提出在数据库参数调优领域应用强化学习
	QTune[91]	DBA设定	Double-state DDPG	模型微调	将查询信息作为状态入调优系统
	DB-BERT[92]	手册提示	Double deep Q-networks	模型微调	利用自然语言模型, 从用户手册等文本中提取提示用于调优
	HUNTER[93]	PCA降维, 随机森林降维	两阶段DDPG	模型微调	使用遗传算法与强化学习结合的方法, 两阶段训练

3.1.1.1 参数筛选

数据库系统包含成百上千个不同的可调整参数, 这些参数影响了数据库的不同方面的性能, 参数之间也会相互关联, 相互影响. 为了减少参数调优算法的搜索空间, 算法需要对数据库参数进行初步筛选. 主要有两种不同的参数筛选方法, 分别是基于经验的策略和基于排序的策略.

基于经验的策略的方法[83−78]借助 DBA 的经验选择可能会影响数据库性能的参数, 这种方法极大地依赖 DBA 的经验的可靠程度. 基于排序的方法[82,93]利用算法衡量各个参数对于数据库性能的影响, 并选择排序中最高的几个作为调优参数.

3.1.1.2 特征选择

与参数筛选类似的, 数据库包含大量的与性能相关的特征, 特征之间也会相互关联, 特征主要包括数据库系统状态和工作负载特征两种.

3.1.1.3 调优模型

基于启发式算法的方法在有限的资源下利用启发式算法或者规则自动调整数据库参数, 这些方法易于实现并且调优速度很快. PGTune 和 PGConfig 是规则匹配的参数调优方法, 这些方法首先设定了一系列用户需要输入的信息, 得到用户输入信息后, 针对每个信息设定一系列判定条件, 从而推断出一组合适的配置. 这些方法具有容易使用, 推荐速度快, 适用范围广的优点, 但是由于考虑的数据库信息有限, 规则模型泛化性不足, 参数粒度粗, 调优效果差. 这些方法还需要大量人工参与, 对于不同场景的通用性也较差.

文献 [78] 利用启发式搜索复用历史调优经验, 利用贝叶斯网络指导算法发现最优配置. 该方法进一步发掘了

配置之间的影响, 不断抽取样本构建新的贝叶斯网络, 循环迭代优化. BestConfig[79]将分流采样法和有界递归搜索方法结合. BestConfig 首先利用分流采样法, 把参数配置空间离散化, 将参数配置变成配置区间, 然后随机选取样本作为参数配置来测试配置性能, 最后找出其中最优的参数区间, 并对该最优空间进行更细致的搜索直到达到理想性能. 这些方法自动化程度有了更加明显的提升, 所得到的参数也更加精确和具有更强的适用性, 但是这些方法推荐速度较慢, 对历史经验利用不充分.

基于贝叶斯优化的方法主要由 3 个组件构成, 分别是基于贝叶斯的调优模型、数据筛选以及模型迁移. 在训练基于贝叶斯的调优模型时, 将数据库参数配置作为输入, 数据库的指标 (吞吐量, 延迟等) 作为标签. 一般化的流程包括: 1) 收集负载、参数和系统状态信息, 2) 参数选择, 3) 将数据输入模型, 4) 参数生成, 5) 结果观测, 然后重复上述 3)、4)、5) 直到达成终止条件.

一些方法[80–82,86,87]使用了不同的配置特征空间选择函数, 这些方法站在不同的角度设定了不同的特征选择. 文献 [80,81] 使用高斯回归作为其调优模型, 高斯回归的预测值是观察值的插值, 一定程度避免不合理的数据库参数配置产生. 其次高斯过程预测值是概率的, 可以计算经验置信区间, 给出产生的数据库参数配置置信度. 贝叶斯模型作为参数调优模型的方法[82–85]能够处理大量数据库参数变量之间的复杂关系, 有效地描述配置之间的依赖关系. 贝叶斯模型也可以根据当前数据库状态对未来的数据库变化进行推断, 所以具有一定的抗干扰能力. 除此之外, 贝叶斯模型容易进行模型迁移, 能够适应数据库变化的场景. 集成的调优模型[86,87], 将多种方法的优点进行结合使得推荐算法的准确性和调优效果有所提升, 从一定程度上提升了调优模型的泛化能力, 模型集成也可以更好地适应不同的数据模式.

基于深度学习的方法无需运行负载而使用深度学习预测给定参数下的数据库性能, 避免调参过程中需要不断运行负载的情况.

IBTune[88]利用缓存未命中率和缓冲池大小之间的关系优化云数据库的内存分配. IBTune 利用数据库相关运行状态找到每个数据库实例可容忍的未命中率, 根据该指标和实例内存来调整目标缓冲池大小. IBTune 还设计了一个神经网络学习数据库状态和参数配置与响应时间的关系. LITE[89]针对 IBTune 不能应对负载差异较大的情况, 针对 Spark 设计了基于对抗学习的自适应系统. LITE 训练了基于神经网络的估计器模型来估计 Spark 在预估参数上的性能. LITE 还使用了迁移学习的方法来应对训练数据不足, 以及生成不同参数和性能的训练数据耗时的情况.

基于强化学习的方法将强化学习的范式应用在数据库参数调优. 数据库系统作为强化学习环境 (environment), 调优器作为代理 (agent) 包括调优算法和调优模型, 数据库指标表示当前状态 (state), 代理将状态 (state) 作为输入, 给出推荐的数据库参数作为当前需要采取的动作 (action), 奖励 (reward) 就是调优配置给数据库系统带来的性能提升, 一般用负载代价减少量或者吞吐量的提升来衡量. 强化学习的代理 (agent) 就是参数调优器, 由调优算法和调优模型两部分组成.

基于强化学习的数据库参数调优方法通过强化学习算法自动化地优化配置, 以提高数据库性能和效率. 基于强化学习的参数调优方法[90,91]多使用人为设定的参数集合作为选择的特征空间. 这主要是因为大多数数据库参数对数据库性能影响有限, 人为设定可以很好地吸取 DBA 的调优经验, 选取对数据库性能影响较大的参数. CDBTune[90]利用 DDPG (deep deterministic policy gradient)[94]算法, 通过模型微调和将查询信息作为状态输入到调优系统中, 实现了对数据库参数的智能调整. QTune[91]采用了 Double-state DDPG[95]考虑了更多的状态信息以进行更精细的调整. 此外, DB-BERT[92]则是利用自然语言处理技术, 从用户手册等文本中提取有用信息, 辅助调优过程. HUNTER[93]使用了基于 PCA (principal component analysis)[96]和随机森林的降维方法对特征空间进行选择. HUNTER 方法则结合了遗传算法和强化学习, 采用两阶段训练策略, 先通过遗传算法进行粗略搜索, 然后用 DDPG 进行精细调优, 结合了对特征空间的探索和利用, 从而平衡全局搜索和局部优化的需求.

基于大模型的方法利用 LLM 的强大能力进行数据库调优, 它们以文本的方式将系统信息和负载信息输入大语言模型以获得推荐参数. GPTuner[97]和 DB-BERT[92]需要人为收集数据库参数设计的文本以限制调优空间, GPTuner 利用 LLM 进行预处理, 也就是提供数据库的参数等的详细信息, 然后让大模型从中选择最相关的参数以对参数空间进行剪枝, 然后利用传统的贝叶斯优化方法进行参数推荐过程. DB-BERT 应用 BERT 模型进行的参数

推荐. 在初始训练阶段, 它利用数据库手册和参数以及历史调优经验微调模型的权重, 以便将自然语言提示为推荐的设置. 在运行时, 利用微调的模型对参数进行排序, 然后利用强化学习来选择最佳的参数.

3.1.1.4 模型迁移

针对新数据变化 (如数据量激增、分布偏移、工作负载动态调整等), 模型迁移方法通过复用历史知识、动态优化参数配置, 显著提升了数据库的适应性和性能.

基于启发式搜索的方法主要依赖“经验转移”机制, 通过进化算法实现跨任务的知识复用, 其特点在于将参数子空间离散化并结合分割采样策略, 利用历史搜索经验优化新任务的参数配置, 但缺乏显式的特征空间建模能力. 贝叶斯优化方法则展现出更强的迁移潜力, 典型如 OtterTune 采用工作负载映射技术, 通过高斯过程回归建立跨数据集的性能预测模型; ResTune 进一步引入元学习框架, 通过预训练调优模型与微调阶段的适应性上下文优化实现跨数据库系统的参数迁移, 其核心在于利用共享的底层特征空间降低环境差异带来的性能衰减. 深度学习驱动的 LlamaTune 采用多模型集成策略, 结合 SHAP 特征重要性解析技术, 将离散化参数空间映射到连续潜在空间以实现跨场景泛化. 与之相比, 强化学习类方法 (如 OnlineTune) 强调动态适应性, 通过在线交互式策略梯度更新 (如 DDPG 算法) 实时响应工作负载变化, 其迁移过程融合了安全性评估机制, 在保证 QoS 约束的同时完成参数调优策略的跨环境迁移.

3.1.2 索引推荐

索引推荐旨在通过分析数据库的数据模式和查询工作负载来自动识别并建议最适合的索引, 从而提高数据库查询的效率, 减少数据库查询响应时间, 优化数据库的存储效率和提高数据库对计算资源的使用率. 通过自动化索引管理的方式, 索引推荐使数据库系统能够适应不断变化的数据访问模式, 从而在不需要 DBA 人工干预的情况下维持高性能.

如图 5 所示, 智能数据库索引推荐的标准流程包含 3 个核心模块. 1) 候选索引生成: 接收负载分析 (查询模式、频率) 和数据库元数据 (表结构、统计信息) 作为输入, 通过启发式方法 (如规则驱动) 和学习式方法生成候选索引集合. 2) 索引推荐模型: 基于候选索引和实时系统状态 (存储、性能指标), 结合离线推荐 (预训练模型) 与在线推荐 (动态调整策略), 采用基于规则、智能算法及强化学习生成最终推荐索引. 3) 索引效益估计: 通过虚拟索引创建 (What-if 索引) 和代价评估 (计算存储、I/O 开销), 量化候选索引对查询性能的提升效果, 为模型提供反馈以持续优化推荐策略. 后文表 4 总结了典型的智能索引推荐方法以及它们分别使用的核心模块.

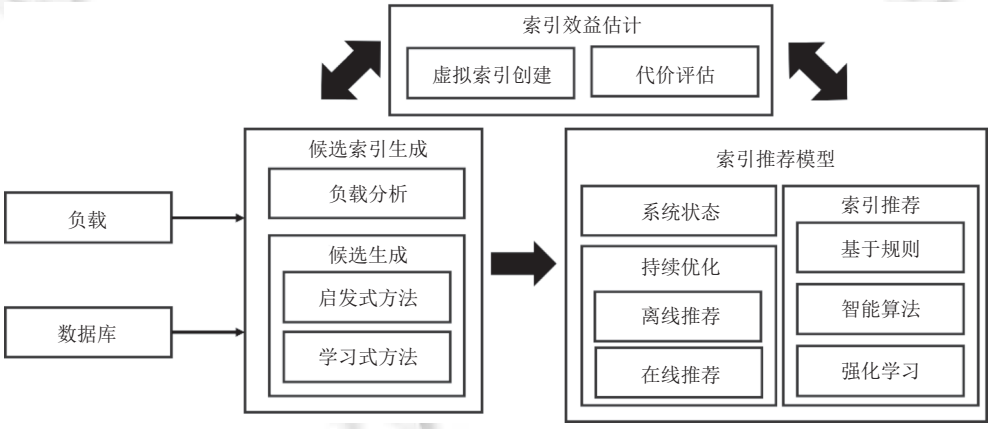

图 5 数据库索引推荐标准化架构

3.1.2.1 候选索引生成

候选索引生成是索引推荐流程的初始阶段, 其核心目标是从数据库工作负载中提取潜在的索引候选集合. 候选索引需满足两个基本条件: (1) 覆盖工作负载中高频查询的过滤条件、连接条件或排序条件; (2) 符合数据库存储空间约束. 此阶段的输出质量直接影响后续索引推荐的效果.

表 4 智能索引推荐方法

方法	方法类型	配置生成	代价评估	终止条件	索引间影响
AutoAdmin[98]	基于规则优化搜索	规则+优化器	What-if+代价推导 (代价减少)	索引数量	隐式
AIMeetAI[99]	基于规则优化搜索	规则	索引成对比较 (代价对比)	存储空间	隐式
BEN_KNAP[99]	基于规则优化搜索	规则+优化器	What-if+代价推导 (单位存储收益)	存储空间	显式
Extend[100]	基于规则优化搜索	规则	What-if+优化器 (单位存储代价减少)	存储空间	隐式
Cophy[101]	基于规则优化搜索	规则	What-if+代价推导 (整数规划)	整数规划	隐式
DISTILL[102]	基于规则优化搜索	规则	What-if+代价模型 (代价减少比)	索引数量	隐式
ISUM[103]	基于规则优化搜索	规则	What-if+代价推导 (代价减少比)	负载数量	隐式
MCTS[104]	强化学习	规则+优化器	What-if+代价推导 (代价减少)	索引数量	隐式
NoDBA[105]	强化学习	规则	优化器 (代价减少比)	索引数量	隐式
LanIdxAdvis[106]	强化学习	启发式规则	What-if+优化器 (代价减少比)	存储空间或索引数量	隐式
COLT[107]	智能算法	规则	What-if+代价推导 (代价减少)	存储空间	隐式
QB5000[108]	智能算法	采样聚类	查询到达率	存储空间或索引数量	隐式
PDAlerter[109]	智能算法	规则	优化器+代价推导	存储空间	隐式
OnlinePT[110]	智能算法	基于计划	优化器 (代价减少)	存储空间	显式
WFIT[111]	智能算法	规则	索引配置工作函数 (代价减少)	索引数量	隐式
AIM[112]	智能算法	规则	What-if+优化器 (CPU成本的代价减少比排名)	索引数量	隐式
LIB[113]	智能算法	规则	代价减少比	索引数量	隐式
DBAbandits[114]	强化学习	规则	总代价	存储空间	隐式
SWIRL[115]	强化学习	规则	What-if+优化器 (单位存储代价减少)	存储空间	隐式
DRLindex[116]	强化学习	规则	What-if+优化器 (代价减少比)	索引数量	隐式
HMAB[117]	强化学习	规则	What-if+优化器 (总时间)	索引数量	隐式
AutoIndex[118]	强化学习	规则	优化器 (代价减少)	存储限制	隐式

索引推荐方法主要分为两种, 分别是传统的组合优化和基于机器学习的方法.

传统的优化组合方法: AutoAdmin[98]通过迭代增加索引宽度生成候选索引集合. 首轮迭代为每个查询生成候选单列索引, 后续通过合并或扩展形成多列索引组合. 采用预剪枝策略, 例如限制候选索引的最大列数 (如 DTA[119]算法初始即考虑多列索引), 并通过种子配置减少枚举次数. Dexter 算法基于查询优化器的假设索引功能, 自动为所有未建立的潜在单列和多列索引 (最大列数为 2) 创建假设索引, 通过优化器反馈筛选候选. DB2 Advisor[120]将所有潜在索引设为假设索引, 直接依赖查询优化器的执行计划推荐候选索引. 优化器在生成查询计划时自动筛选出对单个查询最优的索引加入候选集. Relax 方法[121]初始候选索引由查询优化器为每个查询生成的最佳索引组成, 后续通过转换规则 (如合并、拆分、删除属性等) 动态调整候选集, 逐步降低存储开销. Cophy[101]采用全枚举策略, 将所有可能的单列和多列索引纳入候选集, 通过线性规划模型筛选满足空间约束的最优组合.

基于机器学习的方法基于查询语法解析提取候选列, 通过排列组合生成多列索引, 结合特征工程筛选. 一些方法从查询的聚合函数、WHERE、JOIN、ORDER BY、GROUP BY 子句中提取列, 生成单列候选, 再排列组合形成两列、三列候选索引. 基于强化学习的方法往往将工作负载编码为矩阵, 通过马尔可夫决策过程生成候选索引状态, 依赖历史负载数据训练模型动态调整候选策略.

3.1.2.2 索引推荐模型

索引推荐模型旨在从候选索引集合中选择最优子集, 需同时优化时间效率 (降低查询延迟) 和空间开销 (控制索引存储). 该问题被建模为组合优化问题, 属于 NP-Hard 问题, 需通过近似算法或启发式方法求解.

基于规则的索引推荐方法通常采用规则驱动的策略生成候选索引. AutoAdmin[98]通过贪心算法迭代选择局部最优索引, 并利用假设代价分析 (What-if) 评估索引收益. BEN_KNAP[99]进一步引入显式收益机制以量化索引间交互效应. Extend[100]采用动态搜索空间生成策略, 通过存储-查询代价比隐式编码索引关联性. Cophy[101]则构建整数

规划模型, 通过解空间剪枝实现高效求解. 随着机器学习技术的发展, AIMeetsAI[122]将索引选择转化为分类问题, 而 DISTILL[102]利用神经网络学习负载-索引映射关系以减少优化器调用开销. ISUM[103]通过负载压缩技术提升算法可扩展性. 在强化学习方向, NoDBA[105]首次尝试单列索引推荐, LanIdxAdvis[106]扩展至多列索引并引入贪心因子优化探索策略, MCTS[104]则基于索引单调性设计新型代价评估函数.

基于智能算法的索引推荐方法根据响应机制差异, 智能算法可分为主动预测和反应分析两类. 主动预测方法如文献 [123] 采用时间序列预测负载特征, COLT[107]基于历史窗口预测索引收益, QB5000[108]则利用神经网络预测查询到达率. 反应分析方法强调动态响应: PDAlerter[109]通过代价边界分析触发调优, OnlinePT[110]实时淘汰低效索引, WFIT[111]引入工作函数量化索引收益, AIM[112]则通过克隆环境验证索引有效性. Learned index benefits (LIB)[113]提出端到端代价估计器, 通过注意力机制解决索引交互问题, 并采用迁移学习增强适应性.

基于强化学习的索引推荐方法将索引推荐建模为马尔可夫决策过程, 其中代理 (agent) 根据数据库状态 (state) 生成索引配置 (action), 并以执行代价作为奖励 (reward) 进行优化. DBA Bandits[114]采用多臂老虎机模型线性评估配置收益, SWIRL[115]使用 PPO 算法优化状态编码策略. DRLindex[116]扩展至分布式场景, HMAB[117]通过分层结构支持多配置联合推荐. 蒙特卡洛树搜索方法中, MCTS[104]基于索引单调性提升搜索效率, AutoIndex[118]实现增量式索引管理.

3.1.2.3 索引效益估计

索引效益估计是量化候选索引对系统性能提升的关键步骤, 需综合评估时间收益 (如查询加速) 与空间成本 (如存储占用). 其核心挑战在于准确预测索引对查询计划的影响, 避免“过度索引”或“索引失效”.

基于优化器的代价估计方法直接依赖数据库优化器的代价模型, 通过模拟索引对查询计划的影响, 估算索引带来的执行时间减少量. 在不实际创建索引的情况下, 通过元数据模拟索引的存在, 利用优化器生成假设性查询计划并计算代价差异. 例如, DB2 Advisor[120]通过优化器反馈筛选候选索引. 这些方法对候选索引进行动态评估, 结合历史统计信息 (如选择度、基数) 优化代价预测精度. 这些方法与数据库优化器深度集成, 结果可解释性强, 适用于静态负载场景, 但是优化器的代价模型可能存在偏差 (如错误的选择度估计), 且无法处理复杂索引交互效应.

基于机器学习的代价估计方法通过机器学习算法 (如强化学习、深度神经网络) 从历史负载数据中学习索引收益模式, 预测候选索引对负载的整体提升效果. 这些方法适应动态负载变化, 能发现传统方法忽略的复杂索引组合模式. 但是依赖大量训练数据, 模型可解释性差, 且计算开销较高.

3.1.3 数据库诊断

智能数据库诊断是一种融合人工智能技术与传统运维经验的全生命周期管理方法, 其核心在于通过数据驱动的自动化流程, 实现异常感知、根因定位与优化执行的闭环管理. 这一流程的构建不仅需要覆盖数据库内部的运行状态, 还需整合操作系统、网络设备等全链路指标, 形成立体化的监控与推理体系.

如图 6 是智能数据库诊断的标准化流程, 智能数据库诊断首先从数据库系统采集和监控数据, 然后针对其中存在的异常进行检测, 在获取异常数据后需要根据异常数据进行根本原因定位, 最后提出修复和优化的建议. 最后系统需要不断进行这个过程以实现对数据库系统的不断监控.

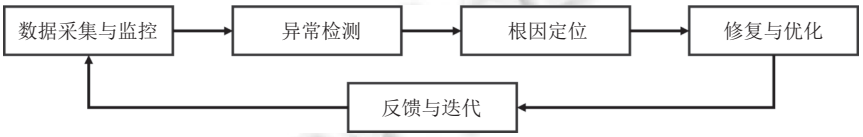

图 6 数据库诊断标准化架构

3.1.3.1 数据采集和监控

数据采集与监控是诊断流程的基石. 系统需实时捕获数据库运行的全维度指标, 包括硬件资源 (CPU 利用率、内存占用率、磁盘 I/O 吞吐量)、查询性能 (响应时间、锁等待时长)、网络状态 (丢包率、延迟) 以及元数据 (索引结构、表分区策略). 例如, PinSQL[124]通过云数据库实例的日志代理采集查询执行计划与资源消耗数据, 而 EGADS[125]则针对大规模时间序列设计轻量级采集框架, 支持千级指标的秒级同步. Twitter 的 S-H-ESD 进一步引

入时间序列分解技术, 将原始数据拆分为趋势、周期与残差分量, 为后续异常检测提供高信噪比输入.

3.1.3.2 异常检测

异常检测与特征提取阶段致力于从海量数据中识别异常信号. 传统阈值检测方法因依赖静态截断值难以适应动态负载, 为此, 文献 [126] 提出的极值理论通过流式单变量分析动态调整阈值, 避免人工干预. Netflix 的 RPCA 采用鲁棒主成分分析, 将高维周期性数据分解为低秩矩阵与稀疏噪声, 精准捕捉促销场景下的异常波动. iSQUAD[127] 则突破二元分类局限, 对异常状态进行细粒度划分, 例如将 CPU 过载细分为“短期尖峰”与“持续饱和”两类, 为根因分析提供丰富上下文. FluxInfer[128]通过构建加权无向图建模指标依赖关系, 结合 PageRank 算法识别核心异常节点, 有效区分因果链中的根因指标与衍生现象.

3.1.3.3 根因定位

根因定位与分析需要从异常信号中追溯本质问题. DBSherlock[129]引入因果推理模型, 将 DBA 经验编码为因果假设库 (如“锁等待激增→事务并发度超限”), 通过概率图模型筛选高置信度根因. ExplainIt[130]则以无监督方式对数千条因果假设排序, 利用贝叶斯网络推断分布式系统中的传播路径, 例如定位跨节点查询延迟的源头. 针对间歇性慢查询难题, iSQUAD[127]设计双阶段分析框架: 离线阶段通过聚类将相似执行计划的查询归类, 在线阶段基于贝叶斯模型关联异常类型 (如 I/O 阻塞或索引缺失) 与慢查询集群, 显著降低人工标记成本. AutoMonitor[131]则发现异常指标的空间聚集特性, 采用加权最近邻算法计算指标相似度, 结合全局权重调整策略提升根因定位精度, 例如在内存泄漏场景中快速识别碎片化严重的缓冲池.

3.1.3.4 修复与优化

修复优化与自动化执行阶段将诊断结论转化为可操作方案. SQLCheck[132]通过解析抽象语法树 (AST) 检测反模式, 例如识别嵌套循环连接导致的笛卡尔积爆炸, 并推荐基于哈希连接的查询重写策略. DBdeo[133]从架构、查询与数据这 3 个层面枚举反模式, 例如检测宽表设计导致的冗余字段, 提出垂直分表方案. DeFiHap[134]在 HiveSQL 优化中融合元数据分析与神经网络预测, 通过连接键值分布与表记录数预测执行时间, 推荐最优化简器 (如谓词下推或分区裁剪). 对于硬件资源瓶颈, AutoMonitor[131]基于 Kolmogorov-Smirnov 检验监控指标分布偏移, 自动调整 InnoDB 缓冲池大小或并发线程数, 使资源配置动态适配负载变化.

3.1.3.5 反馈与迭代

反馈迭代与模型更新形成闭环学习机制. PerfXPlan[135]在 MapReduce 作业调试中持续收集性能数据, 通过在线机器学习更新查询代价模型, 使系统能够适应数据倾斜模式的变化. 文献 [136] 提出的“指纹”技术将历史健康状态编码为特征向量库, 当检测到当前指纹与库中记录的距离超限时, 触发模型再训练以纳入新的异常模式.

3.2 数据库负载分析与管理

数据库负载指系统在特定时间窗口内处理的操作集合, 包含结构化查询、事务操作及并发控制等行为, 其特性受 OLTP 与 OLAP 场景差异的显著影响. 传统数据库管理员 (DBA) 依赖经验主义的负载管理方法已难以应对云原生环境下动态负载的复杂性, 而智能负载管理通过构建预测模型与模式挖掘算法, 实现了从被动响应到主动决策的范式转变.

3.2.1 负载预测

智能负载预测是通过机器学习算法分析数据库历史负载特征与资源消耗模式, 动态推演未来时间段内系统负载强度及资源需求的技术.

数据库负载预测遵循数据采集、模型训练与决策优化的三阶段机制. 系统首先通过时间序列监控模块采集查询吞吐量、CPU 利用率等核心指标, 并整合查询复杂度、资源竞争度等衍生特征构建训练数据集. 继而采用动态集成学习方法, 将时序分解模型、深度神经网络与传统回归算法相结合, 通过在线权重调整机制优先选择预测误差低的模型组合. 最终基于鲁棒优化理论生成资源配置策略, 综合考虑历史误差分布与业务优先级, 实现资源的弹性伸缩.

云数据库环境下, 负载预测通过资源需求建模实现动态资源调配. 文献 [136] 在 Azure SQL 构建的原型系统

采用信号提取技术, 通过多场景实验捕获资源需求特征以实现自动化扩展. 文献 [137] 提出基于时间序列分析的容量规划方法, 通过模式识别与集成建模预测周期性负载. Seagull[138]则利用多模型融合预测低负载时段以优化备份调度. 在大数据领域, TASQ[139]采用参数化性能特征曲线与数据增强技术解决稀疏数据下的资源分配问题. 文献 [140] 通过蒙特卡洛树模拟多租户资源需求相关性, SUFS[141]结合 LSTM 与自适应统计实现突发数据下的鲁棒存储预测. Auto-WLM[142]通过查询执行特征建模实现 Redshift 平台的智能并发控制, 其优先级调度算法可提升 23% 吞吐量.

3.2.2 负载生成

智能负载生成指利用强化学习、生成对抗网络等人工智能技术, 合成符合特定语法规则、语义约束或性能目标的数据库操作序列.

智能负载生成包含约束定义、算法生成与效果验证这 3 个关键阶段. 首先使用声明式语言将业务规则转化为可执行的约束条件, 包括查询结果基数范围、事务完整性要求等核心限制. 随后采用强化学习框架, 在保证 SQL 语法正确性的前提下, 通过奖励机制引导生成器探索符合性能目标的查询组合. 生成结果需通过多级验证体系: 语法层面检查查询结构合规性, 语义层面验证连接谓词逻辑一致性, 性能层面比对执行计划代价估算.

智能负载生成分为约束型与非约束型两类: (1) 约束型生成: LearnedSQLGen[143]采用强化学习框架, 通过基数约束引导的奖励函数与有限状态机确保查询有效性; ezGen[144]通过概率近似模型实现子查询的语义保持生成. (2) 非约束型生成: Lauca[145,146]通过事务逻辑与数据访问分布的双重特征建模实现性能指标复现; DBMS Annihilator[147]采用软硬件协同设计支持百万级事务压力测试.

3.2.3 负载检测

智能负载检测是通过实时解析查询特征与资源消耗轨迹, 识别工作负载分布偏移、性能异常及资源竞争问题的自动化诊断技术.

负载检测系统基于特征分析、异常识别与动态调优的闭环机制运行. 系统首先从查询文本、执行计划和资源指标这 3 个维度提取特征: 语法特征包括操作符类型与子句结构, 语义特征涵盖谓词选择率与连接复杂度, 资源特征涉及 CPU/内存使用波动模式. 继而采用混合检测策略, 对渐进式负载偏移计算历史分布差异度, 对突发异常识别统计离群点, 并通过滑动窗口机制实现秒级响应. 检测结果触发分级响应策略: 常规偏移自动优化索引配置与查询路由, 严重异常则触发告警并保存诊断快照.

工作负载偏移表现为查询分布或数据特征的时序变化. 文献 [148] 的轻量级模型通过实时差异检测触发重配置. 文献 [149] 采用基于距离的聚类分析检测负载演变. 文献 [150] 提出双样本检测框架监测查询特征分布漂移. DBAugur[151]利用 GAN 网络建模负载趋势与异常关联. AWM[152]则通过马尔可夫链与前缀树实现复杂模式发现, 其成本优化模型可大大降低检测延迟.

3.3 小 结

在智能数据库自动化管理和运维领域, 当前研究呈现出 3 个显著的发展趋势: 首先, 基于人工智能的调优算法正深度整合自适应学习机制与强化学习框架, 通过动态适应数据库特性和工作负载模式, 显著提升参数优化与索引推荐的自动化精度. 其次, 新型负载管理模型通过融合多维特征 (包括实时性能指标、历史负载模式及系统状态数据) 构建闭环反馈系统, 实现了亚秒级响应的动态资源分配. 值得注意的是, 可解释 AI 技术在异常诊断中的应用, 通过构建层次化解释模型, 不仅提升了故障定位准确率, 同时生成符合 DBA 认知逻辑的决策依据.

现有研究对数据库参数调优领域进行了系统性的总结与梳理[153−155], 这些研究从技术演进、方法论体系及实现流程等维度对数据库参数调优进行了全面的归纳与分类. 在数据库索引推荐研究方面[156,157], 文献 [156] 采用多维度分析方法, 既对现有索引推荐技术进行了层次化比较, 又系统梳理了该领域的技术发展脉络. 与之相补充, 文献 [157] 则构建了索引推荐的标准化框架, 并基于此对各功能模块进行了横向对比分析.

智能管理层技术 (参数调优、索引推荐、诊断、负载分析) 的演进, 是标准化处理范式在数据库运维自动化领域的典型实践. 这些技术完整覆盖了系统感知 (系统状态监控)、特征提取 (参数/负载特征筛选)、模型应用 (启

发式/贝叶斯/强化学习决策)、决策执行 (配置应用/索引创建) 及反馈管理 (性能监控与模型迁移) 的闭环流程. 在智能数据库管理层的研究和方法中, 针对数据库调优与诊断和负载分析与管理的人工智能和机器学习算法已经取得了显著的进展, 但仍面临一些深层次的挑战. 对于数据库调优与诊断, 虽然现有的算法能够在一定程度上自动调整参数、推荐索引等, 但现有的方法往往需要大量的训练数据, 较长的训练时间和较高的算力支持. 面对不断变化的工作负载和复杂的数据库架构, 现有模型的泛化能力和调优精度仍需进一步提升. 此外, 在人工智能的加持下数据库诊断的自动化程度虽然大幅提升, 但在准确识别数据库问题和详细解释根本原因方面, 这些方法还需要进一步的优化. 对于数据库负载分析与管理, 负载预测算法虽能捕捉到负载变化, 但在动态环境中预测的长期准确性仍然难以保证. 负载生成算法也需要依赖约束条件能够更精准地生成约束负载和更精细地模拟真实负载.

数据库调优与诊断的算法需要进一步结合人工智能算法和数据库特点, 通过自适应算法和强化学习等技术进一步提升调优的自动化水平和准确性. 同时, 数据库开发人员需要考虑如何利用数据、负载和数据库状态信息, 开发出更加灵活和精准的实时反馈的负载管理模型. 此外, 智能数据库需要针对数据库诊断引入解释性模型, 提高数据库诊断的准确率同时向 DBA 提供清晰且可信度高的异常问题解释. 在智能数据库的管理与维护的推荐和优化过程中, 提升算法的透明度和可解释性以增强 DBA 对智能方法的信任和接受程度是算法能够实际应用的关键.

未来突破方向聚焦于: 1) 开发轻量化框架, 通过迁移学习降低模型对标注数据的依赖; 2) 构建时空联合建模的负载预测体系, 整合时间序列分析与图神经网络技术; 3) 建立诊断结果的置信度量化指标, 采用贝叶斯深度学习生成概率化解释. 特别需要强调的是, 任何智能运维算法的实际部署都必须通过人机协同验证 (human-in-the-loop) 机制, 确保技术方案既符合 DBA 的经验认知, 又能突破人工处理的性能瓶颈.

4 智能内核层

数据库内核的智能组件是指利用人工智能的方法优化或者替换集成在数据库系统核心层的高级功能模块, 通过使用机器学习、深度学习或强化学习等方法来增强数据库的自主性和可靠性以及提高数据库性能. 面向内核的智能组件包括数据存取、查询优化和查询执行这 3 个方面.

4.1 数据存取

智能数据库的数据存取仍然依赖传统的数据库读取和存储数据的模式 (读取和存储在内存或者磁盘的方法). 但是数据库社区发现机器学习模型在有序数据读取方面的巨大优势, 利用学习的方法设计了新的索引结构范式来加速数据的读取. 同样的在传统的数据库存储基础上, 数据库社区利用学习的方法设计出了新的数据分区方法加速数据的存取.

4.1.1 学习索引

(1) 一维学习索引

索引是对数据库表中某一列或者多列进行排序的数据结构, 使用索引可以快速访问该列的特定信息. 数据库社区近年来发现使用机器学习模型可以代替传统的索引. 如果将传统的 B-树[158]索引看作黑盒模型, 其输入为查询的键值, 输出为键值对应的存储位置, 就会发现机器学习模型具有类似的功能, 同样将待预测数据输入机器学习模型, 机器学习模型将输出预测数据所对应的标签. 受此启发, 数据库社区希望找到一种学习的模型和数据结构能够代替传统的 B-树, 学习键值和位置之间的映射关系.

给定<键, 位置>对的排序列表, 学习索引旨在使用机器学习模型来预测查询键的位置. 图 7 总结了一维学习索引的标准化框架, 如图所示, 一维学习索引需要支持查询、插入、删除和块加载这 4 种基本功能. 如表 5 所示, 接下来本文将结合具体的方法介绍和总结一维学习索引的这 4 种功能.

查询包括点查询和范围查询. 对于点查询来说, 一般包含 3 个阶段, 分别是层次模型内部递归, 叶节点位置预测和位置修正. 学习索引多为层次结构, 内部模型不断递归预测查询键所属下一层子节点, 为了保证查询正确性叶节点模型需要记录模型的最大误差边界 E. 叶节点模型位置预测就是利用叶节点学习模型预测输入键的位置, 由于模型的准确性有限, 预测的位置往往是不正确的, 所以需要根据模型的最大误差边界 E 对预测的位置进行调整.

对于范围查询, 往往需要先通过点查询预测两个端点位置, 然后扫描两个端点位置范围内的数据返回满足范围查询的键值.

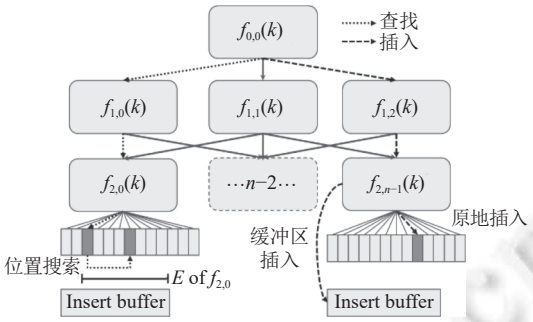

图 7 一维学习型索引标准化架构

表 5 一维学习索引

索引方法	模型	查找		插入/删除		块加载
		节点设置	位置搜索	策略	数据结构	
RMI[159]	简单神经网络/线性回归	无特殊设置	二分查找	无	无	从上到下
XIndex[160]	RMI/分段线性回归	无	二分查找	插入缓冲区	基于误差的节点分裂	从下到上均匀划分
FINEdex[161]	分段线性回归	无	SIMD指令优化的搜索	插入缓冲区	缓冲区满训练合并	从下到上, 贪心策略
SIndex[162]	线性回归	无	二分查找	插入缓冲区	缓冲区合并	从下到上, 贪心策略
ALEX[163]	线性模型	间隙数组	指数搜索	间隙数组原地插入	基于代价的节点合并/分裂	基于代价, 从上到下
MADEX[164]	线性插值	元数据	CDF模型+矫正模型	原地插入	缓冲区满节点分裂	从下到上, 贪心策略
LIPP[165]	非线性模型	条目及位向量	精确位置	原地插入	缓冲区满或冲突, 子树重建	基于冲突分裂从上到下

对应上述查询的 3 个阶段, 影响查询效率的关键因素主要有以下 3 方面, 分别是层次结构本身、节点模型和搜索算法. 首先, 层次结构本身也就是学习索引的结构框架, 包括层级结构设计、节点的设计和额外信息的存储设计. 如 ALEX[163]区分了数据节点和内部节点, 同时在内部节点使用了额外空间存储其子节点信息. 节点模型包括线性模型与非线性模型, 线性模型往往能获得更快的查询速度, 但是查询精度较差, 非线性模型往往具有较好的查询精度, 但是计算代价更高, 非线性模型主要有多项式拟合模型和神经网络模型两种. 非线性模型的另一个优点是支持比线性模型更多的键, 因此单个节点包含更多数据且树的深度小. 除此之外一些方法尝试混合使用线性模型和非线性模型, 例如, XIndex[160]采用两层分层结构, 其中第 1 层使用 RMI[159]模型 (因为第 1 层包含许多键值), 第 2 层节点使用分段线性模型. 一些方法预测完全准确, 无需重新调整预测结果, 如 LIPP[165]. 另外一些方法, 仅在叶节点存在预测误差如 RMI、ALEX, 因此只需要在叶节点模型误差边界范围内搜索. 如果叶节点和内部节点均存在错误, 则需要在叶节点和内部节点均进行修正. 针对不同的误差大小, 搜索算法也不尽相同. 如果搜索误差较小, 线性搜索更为高效, 如果搜索误差较大, 可以使用二分查找的方法, 或者二分查找的变体, 如指数搜索、插值搜索等方法.

插入就是将新的键值插入正确的位置. 由于学习索引需要维护数据有序性, 所以键值插入往往比传统索引更为复杂. 插入新的键值可能会改变索引的部分结构, 或者导致节点模型的重新训练. 一般来说有两种不同的方案支持键值插入, 分别是原地插入和临时缓冲区. 原地插入方法, 如 ALEX[163]、MADEX[164]、LIPP[165]就是找到新插入键值的正确位置, 然后直接将数据进行插入并根据情况判断是否需要对索引结构进行调整. 基于临时缓冲区的插入方法, 如 XIndex[160]、FINEdex[161]、SIndex[162]就是增加一个临时缓冲区, 当缓冲区满就将缓冲区中的数据统一

处理, 临时缓冲区虽然可让学习索引更快捷地插入但是查询时需要对缓冲区数据进行扫描, 缓冲区满或者系统空闲时需要将缓冲区的数据合并到索引, 这带来了查询和合并的额外开销.

删除键也可能会更改索引结构 (例如, 节点合并), 或者模型的重新训练. 键删除的处理方式与键插入类似, 有时为了避免索引结构的频繁改变, 只标记删除的键值 (如 ALEX[163]), 系统空闲或者删除数据到达一定条件再对删除数据进行处理, 删除过程可能涉及索引结构重新调整或节点模型的重新训练.

块加载就是一次性为一批<键, 位置>对构建学习索引. 主要包含有两种类型的方法. 自上而下的方法, 如 RMI[159]、ALEX[163]、LIPP[165], 首先初始化根节点, 然后将根节点拆分为子节点, 并以递归方式处理子节点中的数据. ALEX 使用基于代价函数的方法从上到下按照代价函数最小的方法批量加载构建索引. LIPP 也是从上到下块加载, LIPP 以遇到数据冲突时分裂节点的方式批加载. 自下而上的方法则是将数据拆分为叶节点, 递归从每个节点中提取最小和最大键构建父节点, 直到根节点, 从而构建整棵树. 为了决定如何拆分数据, 有许多算法会考虑拆分开销以有效地构建树结构. 其中 XIndex[160]采取先均匀划分数据, 然后依据数据块组成的叶节点向上构建整棵树的策略.

(2) 多维学习索引

随着大数据发展, 呈几何倍数增长的空间数据的数量和多维数据对数据库查询速度也有了更高的要求. 学习多维索引在学习的一维索引的基础上将其结构进行扩展使其支持多维数据, 同时需要支持特定的空间查询 (如 kNN 查询). 这些方法大多采用将多维数据降维的方法, 然后利用一维学习索引学习降维后的数据和其位置关系. 如表 6 所示, 根据索引的构建方法不同, 本文将现有的多维索引分为基于映射的多维索引、基于空间划分的多维索引和基于格的多维索引, 最后我们总结了一些针对多维学习索引的增强方法.

表 6 多维学习索引

索引方法	类型	ML方法	数据空间	点查询	范围查询	kNN查询	更新/删除
ZM-Index[166]	基于映射	神经网络, 线性模型	填充曲线映射	精确	精确	不支持	不支持
ML-Index[167]	基于映射	神经网络	映射函数	精确	精确	精确	不支持
LISA[168]	基于映射	格回归	映射函数	精确	精确	精确	支持
IF-Index[169]	基于空间划分	线性插值	原本空间	精确	精确	不支持	不支持
RSMI[170]	基于空间划分	神经网络	原本空间 (映射排序)	精确	近似	近似	支持
QD-Tree[171]	基于空间划分	强化学习	原本空间	精确	精确	不支持	不支持
Flood[172]	基于格	分段线性模型, RMI	原本空间	精确	精确	不支持	不支持
Tsunami[173]	基于格	线性模型	原本空间	精确	精确	不支持	不支持

基于映射的多维索引就是使用降维方法将多维数据映射到一维空间, 基于映射值对这些多维数据进行排序, 然后使用一维索引的方法索引这些映射后的数据点. 在查询时, 只需要使用对应的映射函数对数据进行映射, 然后使用学习索引进行查询. 值得注意的是, 为保证查询的正确性, 需要保证映射函数的单调性.

ZM-Index[166]是第 1 个多维学习索引, 该方法选择 Z 阶曲线作为映射函数. ZM-Index 将数据空间划分为网格从而有效快速地计算键的 Z 阶曲线映射值. 为了处理范围查询, ZM-Index 将查询范围分解为 Z 曲线的地址区间, 查询通过索引模型找到相应的网格从而找到对应的存储位置. ML-Index[167]采用改进的 iDistance[174]函数来投影多维数据, iDistance 通常用于索引高维数据以进行高效的最近邻搜索. ML-Index 首先使用聚类方法选定一组参考点, 并将数据依据参考点进行划分, 然后使用 iDistance 方法进行映射和排序, ML-Index 同时使用类似 iDistance 的方法构建 B+树维护 iDistance 值. LISA[168]主要解决了基于填充曲线的多维索引会访问与查询矩形无关的数据块问题. 为了解决这个问题, LISA 采用了基于网格的投影方法. 首先将多维数据划分为等深的格, 将格内的键值使用勒贝格测度与整个格的测度的比值进行映射. 然后使用一个单调的分片预测函数 (格回归) 将每个点映射到其分片 (Shard) 的编号. 最后, 属于同一分片的点被存储到数据页中, 并训练本地模型来定位正确的数据页. 除此之外, Z-Index[175], LMSFC[176], WaZI[177]也是基于映射的学习索引. 这些方法从索引的构建代价, 空间划分或者更新对基于映射的多

维学习索引做了更细致的优化.

基于空间划分的多维索引仍然保留传统多维索引 (如 R 树[178]) 对数据空间的划分方法, 将学习的方法加入递归索引结构中以增加搜索效率. IF-Index[169]用一维学习索引的查询替换了现有树结构 (例如 R 树) 中的叶节点查询过程. IF-Index 的非叶节点遵循普通 R 树构建的构建方法, 但叶节点不仅存储相应的数据页, 还存储模型元数据. 模型元数据包括用于对数据页中的点进行排序的维度和用于预测数据页上的搜索关键位置的线性插值模型. RSMI[170]在叶节点和内部节点中都使用基于神经网络模型进行查询预测. 与 KD 树[179]类似, RSMI 先使用压缩填充曲线对数据进行映射, 然后每层将父节点空间划分为等深单元格, 并训练学习模型将单元格内的数据映射到下一层的划分. 停止划分的条件是当前节点处理的数据数目小于阈值, 最后 RSMI 训练其叶子模型来预测每个点对应的块编号. QD-Tree[171]对特定的数据和查询工作负载进行优化, 利用强化学习对数据进行分区, 以便将特定查询工作负载访问的块数量降至最低, QD-Tree 的非叶节点都使用特定的查询谓词对数据进行分区, 而叶节点中的数据则被划分到同一个磁盘块. QD-Tree 的构造过程可以表述为马尔可夫决策过程, 索引的节点集合表示为状态, 动作空间表示为查询谓词集合, 并使用所有查询中跳过的块数计算奖励. PolyFit[180]和 LearnedKD[181]同样是基于空间划分的学习索引, 它们是将传统的空间索引与学习索引结合的方法, 这两种方法也利用学习模型的优势在经典索引架构的基础上提升查询效率.

基于网格的多维索引的目标是学习紧凑的多维网格以有效地处理正交的范围查询谓词. Flood[172]采用通用格索引作为数据布局. Flood 首先选择排序维度对每个网格单元内的数据排序, 其余维度的数据采用各自维度进行网格叠加. 不同于通常的网格索引, Flood 使用学习的 CDF 模型 (即 RMI[159]) 构造网格分区. 为了更快地细化和过滤查询的范围, 对于每个存储格, 使用排序维度的数据来训练 CDF 模型. Flood 建立了成本模型, 然后使用历史查询工作负载来选择排序维度并调整其超参数. Flood 还设计了索引增强方法以快速处理范围查询, 首先检索与查询超矩形相交的格, 然后查询这些格内的 CDF 模型以应用基于范围查询谓词的有效过滤. Tsunami[173]是对 Flood 的改进方法, Tsunami 主要在处理数据相关性和应对倾斜查询负载两方面做了针对性改进. Tsunami 由适应倾斜查询工作负载的格索引和经过优化以捕获数据相关性的增强网格索引两部分组成. 与 Flood 相同 Tsunami 也是基于选定的维度构建树形索引, Tsunami 将整个多维空间划分为几个不相交的区域, 减少每个区域内历史工作负载的查询偏差, 然后为每个区域构建增强网格. 增强网格充分考虑了历史工作负载, 对频繁查询的区域进行密集分区. SPRIG[182]提出了一个新的多维学习模型, 它是通过采用空间插值和一种新颖的动态编码技术来改进现有的基于格的方法. COAX[183]通过学习数据集属性之间的相关性来降低数据集的维度, 从而使索引空间占用更小、查询更高效.

4.1.2 数据分区

数据分区是一种数据库优化技术, 它将大型数据集按照特定规则划分为多个逻辑或物理子集, 以提高查询性能、简化数据管理并优化存储资源利用. 现代数据库系统中, 数据分区的标准化流程通常包含 4 个关键阶段: (1) 分区策略选择, (2) 分区键确定, (3) 分区方案实施, (4) 动态调整与优化.

在存储模型层面, 数据库物理分区设计主要采用行存储 (N-ary storage model, NSM) 和列存储 (decomposed storage model, DSM) 两种基本方式. 行存储将每一行的数据连续存储, 适合快速整行读写操作; 而列存储将表的每一列分开存储, 有利于加速数据聚合和分析操作. 这两种存储模型分别针对 OLTP 和 OLAP 系统进行优化, 但随着混合事务分析处理 (HTAP) 需求的增长, 出现了融合两者的智能分区方法.

(1) 水平分区

水平分区技术按照数据行进行划分, 主要分为 3 类方法: 基于分区函数的确定性方法、基于外键的启发式方法和基于强化学习的自适应方法. 基于分区函数的方法通过查询谓词将数据元组聚类, 寻找最优分区函数以最小化总体成本. AdaptDB[184]提出的方法根据连接频率进行动态分区, 而文献 [185] 开发了细粒度分区技术使查询能够跳过不相关分区. 这些方法虽然高效, 但在分布式环境中的适应性有限. 基于外键的启发式方法通过分析表间引用关系提高数据局部性. 文献 [186] 采用代价限制的启发式算法选择分区键, 而 Clay[187]系统通过监控工作负载动态识别和扩展“热元组”分区. 这类方法虽然提高了查询性能, 但往往需要承担数据冗余的代价. 基于强化学习的方

法代表了最新研究方向. Neuroshard[188]系统直接将工作负载转化为神经超图, 通过多任务学习优化多个分区目标. 文献 [189] 将表结构和查询特征编码为状态向量, 使用深度强化学习进行分区决策. 这些方法虽然展现了强大的自适应能力, 但在动态工作负载下的评估效率仍有提升空间.

(2) 垂直分区

垂直分区技术按列划分数据, 特别适合处理宽表和混合访问模式. GridFormation[190]框架采用强化学习方法构建了包含代理、环境和动作空间的完整分区决策系统, 支持在线自主调整. 文献 [191] 则针对 JSON 数据开发了轻量级内存关系数据库和专用垂直分区算法, 有效解决了半结构化数据的关系化支持问题. 值得注意的是, 现代垂直分区算法已不再完全依赖工作负载特征, 而是结合数据属性本身进行智能划分. 深度强化学习框架的引入使得系统能够自动学习不同分区方案的成本, 实现更优的权衡决策.

(3) 混合分区

混合分区技术综合了水平和垂直分区的优势, 为 HTAP 系统提供了统一解决方案. HYRISE[192]针对内存数据库优化缓存性能, 通过精确的缓存命中预测模型计算最佳分区. H2O[193]采用基于亲和度矩阵的惰性方法生成分区策略, 其模块化设计能快速适应负载变化. Jigsaw[194]算法采用自上而下的方法, 先分别进行水平和垂直分区, 再按访问模式相似性合并分区. Dalton[195]系统则通过强化学习构建轻量级分区算子, 利用历史分区经验快速适应新负载. Grep[196]采用图模型编码数据和查询特征, 通过图神经网络捕获数据相关性并智能选择分区键. Casper[197]设计了独特的工作负载驱动优化框架, 将分区问题转化为二进制整数优化问题. 这类混合方法不仅考虑了分区布局, 还整合了更新策略和缓冲区管理, 为复杂工作负载提供了全面支持.

4.2 查询优化

查询优化是决定数据库响应用户请求效率的重要因素之一. 本文针对查询优化模块进行了标准化表述, 将其核心流程 (如查询重写、执行计划生成与选择等) 归纳为统一的规范框架.

图 8 是智能数据库查询优化的标准化流程, 查询语句输入数据库后, 其首先被解析为初始的逻辑表达式然后由查询优化器按照预先设定的查询重写规则优化该逻辑表达式. 得到新的逻辑表达式后, 传统查询优化器大多使用动态规划枚举或者启发式算法生成不同的物理查询计划, 通过代价估算对比计划代价确定最终执行的物理计划. 所以最终执行物理计划的性能依赖于代价估算的准确性和计划优化器探索计划空间的有效性. 查询规模估算用于代价估算模型中, 以计算执行计划操作符的成本. 获取最终的物理计划后, 数据库计划执行器将按照计划优化组件所产生的物理计划执行查询并将查询结果返回给用户. 如图 8 所示, 智能数据库优化器利用历史查询负载或者对数据特征建模来提高其输出最优执行计划的能力. 智能模型从数据库和负载相关信息中得到特征输入后, 通过有监督或者无监督的方式完成模型训练. 根据模型不同的设计目的可以选择替换传统优化器中的查询重写组件、规模估算组件、代价估算组件或者替换整个计划优化组件.

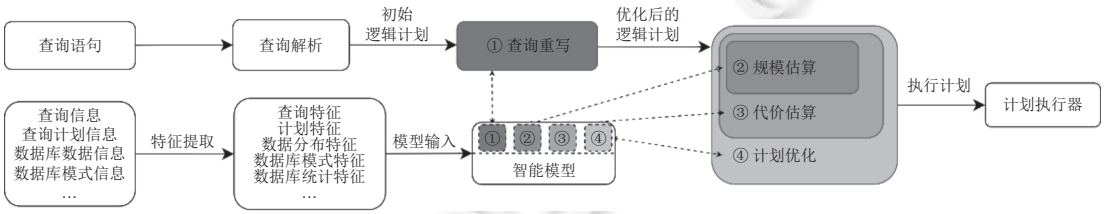

图 8 查询优化标准化架构

4.2.1 查询重写

查询重写是数据库查询处理的优化技术, 查询重写将用户或应用程序提交的查询语句转换成一个或多个在逻辑上等价但在性能上更高效的形式. 查询重写的目的是在不改变查询结果的前提下, 提高查询执行的效率. 查询重写是一个 NP 难问题, 传统的查询重写方法多采用基于规则或启发式算法生成重写规则顺序进行查询重写, 不同的查询需要匹配不同的重写规则顺序以产生最佳的优化结果, 基于规则和启发式的算法往往导致次优的结果.

基于人工智能的查询重写方法, 利用人工智能的方法选择重写规则的顺序, 从而达到重写查询提升查询效率的目的. SIA[198]提出了一种重写查询谓词的方法, SIA 解决了谓词重写的两大问题, 一是重写后谓词与原始谓词等价的问题, 二是约束合成谓词中的列集合导致的错误问题. SIA 利用可满足性模理论 (SMT) 验证重写谓词和原谓词的等价关系, 从而保证重写的等效性. SIA 还设计了一种反例指导的学习方式学习严格有效的重写谓词. WeTune[199]受到了编译器超级优化的启发, 尝试有约束地枚举所有有效的查询重写规则构造新的查询树, 并从中筛选那些可能有效的可行计划. 该方法也使用基于 SMT 问题的求解方法, 对数据库查询重写规则建模为允许编码的 SMT 问题, 以检查查询重写后生成规则的正确性. WeTune 仍然基于数据库本身的代价优化器来计算重写后的查询树与原查询的代价差距. 尽管 WeTune 与基于启发式规则的方法类似, 但其具有优化迅速, 组件可替代, 能够快速部署的优点. LearnedRewrite[200]基于蒙特卡洛树搜索设计了不同的查询重写方法, 该方法能够高效寻找最优重写顺序, 同时评估重写所降低的查询代价. LearnedRewrite 首先将查询重写规则建模成蒙特卡洛树, 树的根节点是数据库生成的查询树, 每个非根节点是通过对其父节点应用单个重写规则获得的重写后的查询树, 而从根节点到叶节点的路径就是对应的重写规则. LearnedRewrite 设计了单独的查询重写代价估计器用来估计重写给查询带来的代价优化. 除此之外, 为了更快速地训练 LearnedRewrie 还设计了并行的蒙特卡洛树搜索方法以提高搜索效率.

4.2.2 规模估算

数据库查询规模估算就是估算数据库查询结果的基数 (cardinality) 或者选择率 (selectivity), 也就是查询操作符生成的结果元组 (即行) 的数量, 或者结果元组数量与表的元组数量的比率. 数据库查询规模估算的准确性是决定数据库代价估算准确性的最重要因素之一. 对于单表估算而言, 查询规模估算受到数据倾斜的影响, 而对于连接 (join) 估算, 查询规模估算又受到关系之间关联性影响. 对于整个计划而言, 整体的规模估算又有可能受到底层节点的误差传递影响. 智能查询规模估算方法就是利用人工智能的方法对查询的基数进行估算, 其可以分为数据驱动的规模估算方法, 查询驱动的规模估算方法和数据与查询混合驱动的规模估算方法.

图 9 展示了智能数据库基数估计的标准化流程, 基数估计管道首先解析输入 SQL 查询以生成逻辑/物理执行计划, 然后从查询谓词、连接运算符和数据分布 (例如直方图) 中提取多维特征, 最后将这些特征输入估计模型 (例如概率图形模型或神经网络) 以预测基数. 其中数据驱动方法学习从数据分布到基数的映射, 而查询驱动方法仅使用查询和计划特征关注计划到基数的关系, 通过解耦特征工程和模型推理实现模块化扩展.

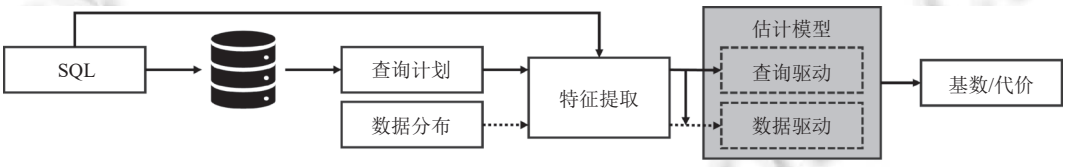

图 9 规模/代价估算标准化架构

数据驱动的规模估算方法主要以无监督学习的方式学习数据的联合概率分布或条件概率分布. DeepDB[201]使用和积网络 (SPN) 拟合数据联合分布, 递归地将一个表的行数据用 sum 节点分为不同的行组、用 product 节点将属性分为不同的列组. 每个叶节点使用统计直方图或分段线性函数估算不同属性数据分布. 在计算查询选择率时, 从叶节点开始自底向上计算, 并根据节点类型累加 (sum) 或相乘 (product) 子节点选择率. 对于多表查询, DeepDB 计算属性的随机依赖系数, 并依据此系数评估多表之间的相关性关系构建和积网络模型并估算选择率. FLAT[202]在 SPN 的基础上提出 FSPN 模型, 它可以支持多属性的统计直方图, 并且在树结构中加入新的节点类型, 增加了 SPN 模型对数据分布拟合的灵活性, 进一步提高了选择率估算的准确性.

而 Naru[203]和 DQM-D[204]则将单表的联合分布用链式法则分解为条件概率分布. 他们使用深度自回归模型例如 MADE[205]拟合条件概率分布, 这类方法可以直接对点查询的选择率进行估算. 对于范围查询, Naru 采用渐进采样策略, Naru 根据条件概率分布的每个内部输出逐列采样, 在深度自回归模型的引导下, 采样器将更加关注查询区域中高影响的部分, 输出用重要性加权补偿诱导的偏差; DQM-D 采用多阶段采样策略, 在每个阶段, 该方法根

据前一阶段的结果, 按其对查询基数的贡献比例选择样本. NeuroCard[206]是对 Naru 的拓展, NeuroCard 在所有表的完整外部连接上构建一个深度自回归模型. 同时 NeuroCard 对大基数列引入了无损列分解, 并使用连接计数表来支持对表子集的任何查询. FactorJoin[207]利用连接直方图高效处理连接操作, 并结合神经网络方法捕捉属性之间的相关性. FactorJoin 将连接查询转换为单表数据分布上的因子图, 并基于因子图模型融合所建立的单表数据分布模型. FACE[208]采用基于标准化流 (normalizing flow) 模型学习数据点的联合概率分布, 将连续随机变量的复杂分布转换为简单分布并计算每个元组的概率密度. SAM-CE[209]考虑部分数据域的稀疏性以及在处理范围查询时数据采样的样本质量对估计误差的累积传播, 提出了一种随机平滑自回归基数估计器. 针对数据稀疏性问题, SAM-CE 向原始分布中添加适量的噪声以平滑数据分布, 使之更加容易学习数据分布, 同时提出了一种平滑采样策略, 以减少由于误差传播而导致的累积误差, 从而提高采样质量.

查询驱动规模估算的方法以历史查询负载作为训练数据, 以有监督学习的方式训练一个从查询语句到基数估计的映射模型. MSCN[210]提出用多组神经网络模型将输入的查询编码为多个特征向量, 再由多层神经网络分别将特征向量转化为特征向量表示, 最后整合这些向量并将其映射为查询的基数大小. Fauce[211]用表的连接图表示数据库中表之间的依赖关系并且基于全局列关系表征各个列之间的依赖关系, Fauce 还定义了数据和模型的不确定性, 并使用训练查询样本重新采样或重新训练的方式减少不确定性. LPCE[212]分别设计了基于 SRU (simple recurrent unit) 模型的 LPCE-I 和 LPCE-R 模型, 前者用作查询驱动的基数估计器以辅助初始查询计划的生成, 后者用于在初始查询计划执行过程中及时根据已被执行节点的真实基数重新优化未被执行的计划节点. 为了提高初始计划的生成效率, LPCE-I 模型采用了知识蒸馏技术, 从一个复杂的 teacher 模型训练 student 模型用于快速输出节点的基数估计.

虽然查询驱动的方法相对于数据驱动的方法有着更快的模型推理速度且支持更复杂的查询类型, 但是当查询或者数据发生改变时, 查询驱动的方法的准确性将显著降低. 因此为了使得查询驱动的方法能更好地适应数据库负载的变化, 一些研究在现有方法之上增加能够增强模型鲁棒性的组件. 针对查询负载的改变, Warper[213]基于 GAN (generative adversarial network) 的思想, 分别设计了生成器合成具有新的查询类型的训练样本和对抗鉴别器区分合成查询和实际观测到的查询, 这使得合成的训练样本集合更加符合查询负载的改变情况, 使模型可以预先适应负载的可能的变化. 对于数据的改变, Warper 基于主动学习的思想挑选有价值的样本以重新获得标签用于重训练基数预测模型. 文献 [214] 在训练阶段隐藏一部分查询特征以强迫模型在特征缺失的情况下学习, 增强了查询驱动方法的鲁棒性. 这也使得即使在有查询变化的情况下, 查询驱动的方法也能快速适应负载变化. 同时该方法通过添加传统优化器的基数估计值作为模型输入特征, 隐藏部分其他查询特征也迫使模型更加专注于矫正传统优化器的原始估计值, 因此该方法可以被视为传统优化器基数估计的矫正方法.

数据与查询混合驱动规模估算的方法就是将查询驱动和数据驱动方法结合. UAE[215]利用 Gumbel-Softmax 技巧来区分类别抽样变量, 使得深度自回归模型可以直接从查询中学习联合数据分布. 因此, UAE 可以采用统一的深度自回归模型, 以无监督的方式学习表的联合分布, 并以监督的方式训练使用查询内容作为辅助信息. ALECE[216]以数据库所维护的表数据的统计特征作为数据特征, 将其输入到注意力机制模型得到数据的表征后, 联同查询特征一起输入到另一个新的注意力机制模型, 最终使用全连接神经网络将表征特征映射为基数估计值. 这种设计增强了模型的鲁棒性, 使得模型能够适应负载的变化.

4.2.3 代价估计

查询代价估算是指查询优化器估算查询计划在当前数据库中的执行时间. 查询代价估算不仅对查询计划的选择有直接指导作用, 还可以用在数据库资源管理等方面. 代价估算不仅与查询计划本身有关, 还受到系统硬件和数据库配置的影响. 传统的代价估算模型通常是专家设定的多项式函数, 这种代价估计模型虽然估计速度快, 但是不能准确反映查询的代价以及系统 I/O 和 CPU 的代价. 智能代价估算就是利用人工智能的方法估算查询的代价. 代价估算的流程同样如图 9 所示, 智能代价估算首先对计划特征进行提取, 然后将计划特征编码作为代价估计模型的输入, 最后代价估计模型输出预测代价.

DNN[217]以计划节点特征作为代价预测的输入同时也考虑了查询计划树的结构特征. DNN 为每类操作符训练

一个神经网络单元, 该网络除了输出当前操作的执行时间, 还输出一个向量, 这个向量和该操作符的父节点特征共同作为父节点模型的输入. Neo[218]、E2E[219]和 QueryFormer[220]分别设计了更加复杂的查询计划编码方式和基于树形结构的神经网络模型. Neo 基于树卷积神经网络模型使用三角形滤波器 (父节点、左子节点、右子节点) 在查询计划树上滑动, 使得它们无需递归步骤即可捕获物理计划树的父子依赖关系. 这样的设计使得模型可以并行处理节点特征, 加快训练速度. 但由于树卷积具有较小的感受野, 每个节点只能从附近的邻居中看到特征, 所以 Neo 无法捕获从叶节点到根节点的长路径信息流. E2E 使用 Tree-LSTM 模型, 将信息自底向上从叶节点聚合到根节点, 并使用最终的输出向量作为物理计划的表示, 最后由多层全连接网络映射为计划的代价估计. QueryFormer 额外添加了数据的采样信息和直方图信息特征, 并分别为这些特征单独编码, 然后使用基于注意力机制的树形神经网络进行代价学习, 这种方法一步改进了神经网络在学习查询计划树形结构特征向量时的信息传输机制, 使得模型能够关注到计划的全局结构特征. Zero-shot[221]通过选择计划中可迁移的特征增强了模型的泛化性, 该方法使用前 n 个数据库上的查询负载作为训练集, 在第 n+1 个数据库上进行查询代价预测. Zero-shot 的输入特征中包含计划节点的基数估计, 因此需要依赖基数估计模型. ParamTree[222]通过建立 C-Param (数据库参数、硬件配置以及计划操作符特征等) 到 R-Param (数据库查询优化器参数中代价函数的各权重值) 的映射函数使得传统代价函数估计器的估计更加准确.

4.2.4 计划优化

将查询解析得到逻辑计划后, 查询优化器将基于计划树的各个节点代价估计输出最终物理执行计划. 生成物理执行计划的过程包括表连接 (Join) 顺序优化和对物理算子选择 (如选择连接算子、扫描算子等). 连接顺序对查询计划性能的影响显著, 一些方法只关注连接顺序优化. 更完整的计划优化方法考虑到整体物理查询计划的影响, 这些方法构造了端到端的查询优化器.

(1) 连接顺序优化

查询计划中表的连接顺序显著影响了查询的性能, 然而寻找最优连接顺序是一个 NP-Hard 问题. 传统的方法主要是基于动态规划或贪心算法等启发式算法选择计划连接顺序, 智能连接顺序优化方法将构造计划连接顺序的问题建模为马尔可夫过程并使用强化学习方法解决. 智能连接顺序优化方法从独立的单表开始自下而上连接子计划和计划中的表以生成完整的连接顺序.

Rejoin[223]采用基于策略的强化学习方法. Rejoin 训练策略网络直接预测下一步动作, 并收集计划代价和状态向量来更新策略网络参数. 一些方法[224−228]使用基于值的强化学习方法, 主要范式为训练价值网络 Q 来预测计划或子计划的未来最小执行时间或代价估计, 然后使用该预测值并基于启发式策略引导完整连接顺序的生成. DQ[225]使用贪心策略, 根据 Q 网络的输出只选择当前最有利的动作. RTOS[227]则进一步改进 Q 网络, 使用 Tree-LSTM 网络表示连接顺序的状态, 同时通过改进训练阶段的损失函数 (包括代价损失函数和延迟损失函数) 来平衡生成连接顺序的时间和效果. JOGGER[228]根据数据库主键-外键关系构建模式图以捕捉表之间的相关性, 同时利用图卷积网络对查询图进行编码, 并设计一个基于树的注意力机制模块对连接计划进行编码. JOGGER 还按照查询的复杂程度分级训练 Q 网络, 采用分层学习的方法来加速 Q 网络的收敛.

(2) 端到端查询优化

端到端的查询优化更加关注计划的整体性能, 是从查询到生成查询计划的完整优化方法. 本文根据执行计划生成方式的不同将现有技术分为自下而上的查询优化方法和规则引导的查询优化方法. 自下而上的优化方法利用机器学习技术完全取代传统查询优化器. 规则引导的查询优化方法旨在充分利用传统查询优化器的专家知识, 通过外部规则的限制引导传统优化器生成更好的查询计划.

自下而上的查询优化方法仍基于强化学习算法框架, 这些方法在 DQ[225]方法的基础上进行拓展, 也通过价值网络的引导以自下而上的方式逐步完成计划构造. 不同的是, 他们不仅优化了连接顺序, 还关注了物理算子的选择, 包括连接算子和表操作符. Neo[218]首先从历史查询负载中学习初始化价值网络, 然后实时地根据输入的查询和其执行结果更新价值网络. 对于输入的查询, Neo 从初始状态 (各个独立单表) 枚举所有可行动作 (包括指定表扫描算子或用某连接算子连接两表) 并得到所有子状态, 再根据价值网络对子状态的预测代价将各个子状态放入一

个最小堆中. 然后, 它将从所有子状态中选出预测代价最小的状态, 作为下一个起始状态继续重复以上操作, 直到得到最终执行计划. Balsa[229]在 Neo 的基础上进一步拓展, 在初始阶段使用简单的自定义代价函数初始化价值网络, 这避免了具有灾难性代价的计划影响整体训练效率. 与 Neo 类似, Balsa 使用预先学习的价值网络引导计划生成. 不同的是, Balsa 每次都会存储预测代价最低的 k 个计划以备后续计划探索. LOGER[230]充分利用了专家经验, 不再像 Neo 或 Balsa 那样直接指定连接操作符, 而是限制操作符的选择, 这在一定程度上减少了动作空间. 在探索策略方面, LOGER 结合了贪心搜索和波束搜索方法, 为每一步动作维护了两个集合包括利用集和探索集. 利用集存储由价值网络估算出的多个最优计划, 而探索集存储上一步未被选中的计划以及从所有候选计划集中随机选取的若干计划.

规则引导的查询优化方法充分利用了现有数据库管理系统的专业知识, 通过预设或生成的规则集合引导传统优化器生成更优的候选计划集合. 这类方法利用历史查询负载训练计划执行时间的预测模型, 用来从候选计划集合中选出最终执行计划. 相较于自下而上的方法, 这种方法在传统查询优化器的保证下训练效率更高. Bao[231]通过专家筛选出的规则集合引导传统查询优化器生成候选计划, Bao 的每个规则集合包含一组操作符限制规则比如禁用循环嵌套连接等. HybridQO[232]通过生成前几个表的连接顺序来引导查询优化器选择更好的计划, 然后使用蒙特卡洛树方法筛选出置信度最高的前缀连接表, 并将其作为引导规则传递给传统查询优化器, 强制其按照所指定的连接顺序生成查询计划, 计划中除了指定的连接顺序外, 其余部分将由查询优化器补充. 在 Lero[233]中, 候选计划是利用启发式算法纠正传统优化器所生成的原始计划的节点基数估计值生成的, 然后其使用 Learning-to-Rank 方法两两比较候选计划代价并选择出最优执行计划. AutoSteer[234]则在 Bao 的基础上, 设计了启发式算法自主地探索规则空间并生成合适的规则集合, 这弥补了 Bao 需要专家经验设计预设规则集合的缺点. FASTgres[235]也基于 Bao 的规则集合设定, 不同的是 FASTgres 选择直接根据输入的查询语句预测合适的引导规则, 再用引导规则让传统优化器直接生成执行计划, 这可以减少查询的响应时间. Eraser[236]聚焦于增强现有方法的鲁棒性, Eraser 采用两阶段策略来确定每个候选计划的预测准确性, 其中第 1 阶段定性过滤所有预期效果不佳的高风险计划, 第 2 阶段定量评估剩余计划的预测质量.

除了以上两种方式外, Leon[237]尽可能地保留了传统优化器的专家知识, Leon 遵循传统优化器的计划生成流程, 用一个由传统代价函数初始化的校准网络模型取代了传统代价模型, 还采用了另一个神经网络模型来帮助传统优化器减少探索不必要的计划空间. 这两个网络模型有效地平衡了计划生成的效益. FOSS[238]则基于强化学习算法框架训练了一个计划生成器对传统优化器生成的计划逐步进行细粒度优化 (包括交换两个表的连接顺序或更换更合适的连接算子), 并训练了一个计划对比器用以从计划生成器所生成的候选计划中选择最终执行计划. 同时为了提高计划生成器的训练效率, FOSS 还利用计划对比器和传统优化器设计了高效的模型模拟训练方法, 这种方法可以通过与计划生成器的快速交互产生大量高质量经验样本用来训练模型.

4.3 查询执行

查询执行作为数据库系统的核心功能模块, 其核心任务是将优化后的查询计划转化为物理操作并生成最终结果. 在智能数据库体系中, 查询执行已突破传统静态执行模式, 演变为包含动态反馈与自适应调节的闭环优化系统.

如图 10 所示, 智能数据库查询执行流程由 5 个核心阶段构成: 物理算子生成阶段负责将逻辑查询计划映射为可执行的物理操作序列, 这是执行引擎的初始化过程; 运行时监控阶段通过性能计数器与采样机制持续采集数据分布特征、资源利用率等运行时指标, 为后续优化提供决策依据; 自适应查询处理阶段基于监控反馈动态重构执行策略, 实现查询计划的热更新; 并发控制与调度阶段采用智能算法协调多任务间的资源竞争, 确保系统吞吐量与响应时间的平衡; 一致性交付阶段通过事务管理机制保障结果集的正确性与可见性, 完成查询生命周期闭环. 这 5 个阶段形成的环形架构通过持续迭代优化, 使系统在动态负载下保持执行效率与资源利用率的平衡提升. 本文后续将重点剖析自适应查询处理与并发控制与调度两个关键环节. 这两个阶段构成了智能数据库区别于传统系统的核心技术特征, 前者通过在线学习实现执行策略的动态调优, 后者借助机器学习突破静态调度策略的局限性.

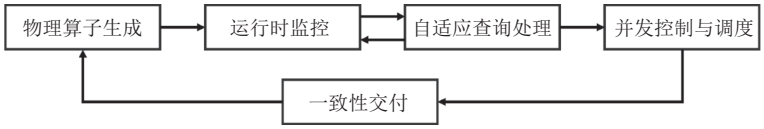

图 10 查询执行优化标准化框架

4.3.1 自适应查询处理

在动态优化环节, 自适应查询处理 (adaptive query processing, AQP) 技术通过运行时反馈机制重构查询计划. Cuttlefish[239]采用多臂老虎机强化学习模型, 在分布式环境 (如 Spark) 中动态探索物理算子组合策略, 其优势在于无需预定义算子调整规则即可实现图像卷积、正则表达式匹配等异构操作的联合优化, 但存在反馈延迟导致优化滞后的问题. RouLette[240]针对多查询场景设计了全局优化器, 通过构建共享运算符的代价模型生成最小化总成本的执行计划, 其强化学习机制有效降低了适配开销, 但在高并发场景下可能遭遇状态空间爆炸的挑战. SkinnerMT[241]则融合了并行计划探索与数据并行处理, 采用元组级线程分配策略实现细粒度负载均衡, 其混合执行架构降低了系统延迟, 但对硬件资源碎片化场景的适应性仍需提升.

4.3.2 并发控制与调度

在资源协调阶段, 智能调度系统通过机器学习模型优化并发控制与任务调度. Decima[242]率先将深度强化学习引入 DAG 任务调度, 通过对历史负载特征的图编码学习生成最优调度策略, 但缺乏对动态到达任务的实时响应能力. 为此, LSched[243]融合图注意力机制与数据树卷积, 构建了白盒化调度预测模型, 可实时感知物理计划特征与资源状态变化, 其查询间/查询内双重调度机制使内存数据库的吞吐量大幅提升. SmartQueue[244]针对存储层优化提出深度 Q-learning 调度框架, 通过神经网络建模缓冲区状态与查询数据访问模式的关联关系, 其自适应排序策略使缓存命中率大幅提高. 针对异构计算场景, BG3[245] 提出了基于 BW 树的内存索引的图存储引擎结合一种工作负载感知的空间回收机制, 提高了存储利用率并减少了写放大. PCC 协议[246]创新性结合持久化内存特性与 RDMA 技术, 通过设计持久化摘要, 从而使流算法能够回答有关流在任何先前时间的查询.

4.4 小 结

数据库内核正通过人工智能方法不断进化, 变得更加智能和高效. 智能数据内核会更加集成化, 将数据处理、分析、存储和检索等功能集成在一个统一的框架内, 同时保持模块化, 便于根据不同需求进行定制和扩展. 随着技术的发展, 智能数据内核将能够处理更大规模的数据, 并实现实时或近实时的数据分析, 以支持快速决策. 内核将具备更强的自适应能力, 能够根据数据的变化和用户的需求自动调整分析策略, 同时通过自学习不断优化性能.

现有研究对智能数据库内核的多个关键方向进行了系统性探讨. 文献 [247] 通过抽象一维学习索引的标准流程, 建立了方法论框架, 并对现有技术方案进行了横向对比分析. 文献 [248] 采用攻击注入的实验方法, 深入考察了一维学习索引在极端场景下的鲁棒性表现. 在理论层面, 文献 [249] 通过数学建模揭示了学习索引的效率上限. 针对多维学习索引领域, 文献 [250] 系统梳理了该技术的发展脉络与演进路线, 而文献 [251] 则通过构建统一评估基准, 对现有多维学习索引方案的性能特征进行了实证研究.

也有一些研究对智能数据库查询优化领域展开了多角度的系统性探索. 文献 [252] 通过实验评估了不同执行计划表示方法在基数估计、查询优化等关键任务中的性能表现. 文献 [253] 基于大量实证研究发现, 当前数据库系统采用的基数估计方法存在显著误差, 而准确的基数估计对确定最优连接顺序具有决定性影响. 在学习的基数估计方法方面, 文献 [254] 构建了系统的设计空间探索框架, 并对现有最优学习方法进行了全面对比分析. 文献 [255] 则从连接顺序选择、访问路径确定和物理操作符选取这 3 个核心维度, 对学习型成本模型进行了综合评估. 为进一步规范评估流程, 文献 [256] 提出了新型端到端基准框架, 为学习型查询优化器的性能评估提供了标准化解决方案.

智能内核层组件 (数据存取、查询优化、查询执行) 的设计严格遵循标准化处理范式. 其流程涵盖数据分布感知、查询特征提取、机器学习模型应用 (如索引预测、基数估算、计划优化)、物理操作执行以及部分增量更新反馈机制.

对于数据获取与存储, 学习索引和数据分区等智能化技术的应用证明了人工智能在优化数据存取方面的潜力. 尽管如此, 学习索引在构建过程中仍面临时间消耗长和计算资源需求大的挑战. 多维索引在提升查询效率和准确性方面尚有较大提升空间. 此外, 数据分区策略在动态环境中的适应性也需进一步研究.

在查询优化方面, 包括查询重写、规模估算、代价评估和计划优化等技术, 已经借助人工智能方法超越了传统的启发式规则. 然而, 这些技术在处理复杂查询和动态工作负载时仍显不足, 如何在动态和复杂环境中准确估算查询规模和成本、生成最优执行计划, 以及如何使人工智能模型快速适应负载和数据变化, 是当前查询优化领域面临的主要挑战.

在查询执行方面, 自适应查询处理、并发控制和查询调度等智能化方法的探索是积极的尝试. 自适应查询处理需要在执行过程中能够更灵活地调整策略, 以应对数据和工作负载的变动. 同时, 在确保系统稳定性和效率的基础上, 还需要更高效地管理资源和任务, 以及进一步智能化并发控制和查询调度.

数据库内核的智能组件模型目前还无法实现实时学习和调整. 数据库领域仍需开发能够在查询执行过程中实时学习和自适应调整的模型, 以优化查询和执行过程. 尽管数据库内核的各个模块都具备智能化的潜力, 但如何实现这些模块之间的有效协同仍是一个待解决的问题. 需要开发一种统一的方法来管理数据存取、查询优化和查询执行等组件, 全面考虑它们之间的相互影响, 以构建一个整体优化的智能数据库内核. 同时, 提高组件所使用的模型的可解释性和透明度也是必要的, 这将帮助数据库开发人员更好地理解模型的决策过程, 从而增强对智能化组件的信任和使用.

5 智能数据库开发接口

大多数智能组件直接集成于数据库中, 或通过特定编程方法实现. 智能数据库开发接口为智能组件的实现提供了统一的抽象接口, 充当了数据库各模块与人工智能方法之间的桥梁. 该接口提供了一系列标准化的编程接口, 便于人工智能方法接入数据库系统, 使智能开发人员无需深入数据库底层开发. 此外, 智能数据库开发接口还支持智能算法开发人员从数据库中获取训练数据并进行模型评估. 目前, 已知的智能数据库开发接口主要有 Database Gyms[257]和 PilotScope[258]两个, 本节将对此进行详细介绍.

5.1 Database Gyms

Database Gyms[257]是一个智能数据库集成环境, 它为数据库组件和智能组件获取训练数据提供了统一的接口. 不同于其他学习领域自定义模拟环境, Database Gyms 使用 DBMS 本身来创建用于机器学习训练的模拟环境, 简化了智能组件模型训练和评估. 在 Database Gyms 中, 智能数据库系统被抽象为 3 个主要实体: 数据库环境、智能代理和用户.

5.2 PilotScope

PilotScope[258]充当智能模型与数据库系统的中间件, 其编程模型显著降低了将智能组件整合到数据库系统中的复杂性. PilotScope 由智能组件到数据库的驱动器和数据库交互器构成. 智能组件到数据库的驱动器包括收集训练数据、训练模型, 并做出决策并优化数据库的标准工作流. 数据驱动器则避免了智能组件对数据库底层的修改, 对不同的数据库, 数据库交互器需要由数据库开发人员以不同的方式实现. PilotScope 抽象出用于机器学习和数据库之间数据交流的方法, 通过这些方法数据库能够收集各种类型的数据、注入智能数据库组件并执行数据库操作.

5.3 小 结

智能数据库开发接口的核心功能是支撑标准化范式中各环节的衔接. Database Gyms 和 PilotScope 等工具抽象了感知环节的数据采集、模型应用的决策注入、执行环节的操作触发及部分反馈管理功能. 这一接口的设计初衷是解决智能技术在数据库系统集成时遇到的部署难题. 未来, 这些开发接口预计将扩展其兼容性, 以支持更广泛的数据库系统, 并形成一套更加通用的人工智能对接机制. 此外, 这些接口将聚焦于提升效率、简化使用流程, 并

实现与数据库系统的深度集成. 开发接口需要在设计上采取更为灵活和开放的策略, 以增强其对不同数据库系统的适应能力.

智能数据库开发接口在发展过程中仍面临若干挑战: 1) 通用性问题: 当前的智能数据库开发接口在通用性方面存在局限, 主要表现在它们往往只能适配特定类型的数据库系统. 例如, PilotScope 目前仅与 PostgreSQL 和 Spark 兼容, 这种局限性限制了其在更广泛数据库环境中的适用性. 2) 完整性考量: 尽管这些组件提供了必要的数据库接口, 但它们在对智能方法的全面评估机制方面尚显不足. 一个有效的评估体系对于确保智能算法的准确性、可靠性和效率至关重要.

6 总 结

本综述深入讨论了智能数据库系统的前沿问题, 以各研究领域的标准化框架视角多维度对比和总结了现有的智能数据库系统、组件开发接口、智能交互层、智能管理层和智能内核层等多方面的最新进展与存在的挑战. 本综述以标准化为核心视角, 系统性地提炼了智能数据库 3 大核心领域 (交互、管理、内核) 内在的通用处理范式. 通过深入分析, 我们揭示了驱动智能数据库自优化的标准智能处理逻辑. 建立这种标准化的理解和描述框架, 为研究者提供了统一的学术分析工具, 促进了不同技术间的对比与模块化设计, 并清晰地揭示了当前面临的数据质量、集成复杂性、安全保障和多模块协同等共性挑战根源在于该标准化流程的不同环节. 标准化框架的建立, 旨在将复杂系统中的多样性抽象为可复用的通用模式, 作为长期演进的基础, 为未来技术迭代 (如多模态查询、动态负载适应、异构硬件加速) 提供可扩展的底层支撑.

更加具体的, 通过对现有智能数据库系统的横向深入分析, 本文评估了人工智能方法在提升数据库与用户交互体验, 帮助 DBA 简化管理流程、提高数据库内核性能方面的优势与不足. 更具体的, 在交互层, 自然语言到 SQL 的转换和表格问答显著降低了非技术用户的门槛, 但复杂语义理解和多模态查询 (如图文混合检索) 仍有待突破; 在管理层, 基于机器学习的自动化调优 (如索引推荐、负载预测) 减轻了 DBA 的经验依赖, 然而极端数据分布下的策略稳定性、实时响应能力仍需提升; 在内核层, 智能组件 (如 learned indexes、基数估计模型) 通过替代传统启发式算法, 在 I/O 优化、查询计划生成等方面展现了性能优势, 但其动态适应性 (如负载突变时的模型迁移效率) 和资源效率 (如 GPU 内存占用) 仍是瓶颈. 值得关注的是, 工业界已涌现出部分成熟方案, 但其黑箱决策机制与数据库事务强一致性需求的矛盾, 揭示了技术落地的复杂性.

各领域当前挑战可归纳如下.

智能交互层在复杂语义解析与多模态交互中存在局限. 模糊查询依赖静态词典导致泛化性差; 跨模态查询缺乏统一的向量化表示与优化器支持.

智能管理层在动态负载与极端场景下的稳定性不足. 时序敏感型任务要求实时响应, 而基于批量训练的模型难以快速适应; 黑箱决策机制阻碍 DBA 对异常操作的根因分析.

智能内核层的组件局部优化与全局效率存在冲突. 学习索引可提升单表查询速度, 但跨表连接时因缺乏协同可能加剧锁竞争; 查询计划生成面对复杂查询和动态工作负载时, 仍然难以生成最优计划; 自适应查询处理和并发控制策略在并发场景下需要更高效可靠的算法来保持系统的高性能.

智能数据库开发接口在全面性与易用性上面临双重挑战. 接口功能碎片化 (如数据采集、特征工程、模型部署模块割裂) 导致开发效率低下, 且缺乏统一的隐私保护机制 (如联邦学习支持不足), 难以满足金融等场景需求. 未来各领域研究需向多层次融合演进.

面向用户的便捷交互需要融合领域知识图谱增强语义理解, 采用增量学习动态更新语义模型; 设计多模态统一查询引擎, 将文本、图像等非结构化数据编码为可联合优化的向量空间.

智能管理层需要构建在线学习运维框架, 通过流式数据处理实时更新模型; 开发可解释性工具, 可视化 AI 决策逻辑; 设计人机协同接口, 确保 DBA 对关键操作的最终控制权.

智能内核层的组件需要推进跨组件联合优化, 例如将查询优化器与并发控制器的代价模型统一为深度强化学

习的联合动作空间; 研发轻量化嵌入式模型, 减少 GPU 内存占用.

数据库开发接口需构建标准化接口框架, 支持动态模型适配、安全数据管道、跨平台部署, 同时提供可视化调试工具, 降低 AI 组件集成门槛.

尽管当前智能数据库的各个研究领域已初步建立起标准化流程框架, 但是, 面对人工智能技术的动态演进与复杂应用场景的膨胀, 突破现有框架已成为必然. 这主要因为: 标准化框架预设的静态模块隔离规则难以兼容 AI 驱动的技术现实. 未来智能数据库的发展方向可能会集中于从“数据库+AI 插件”的松散耦合转向 AI 原生设计. 数据库社区需要开发便捷查询、易于维护和快速响应的完全由人工智能驱动的数据库. 人工智能的技术现阶段需要提升其在数据库系统中的适用性和准确性, 以更好地应对动态变化的数据和查询. 包括开发更准确高效的模型、深化自然语言理解能力、增强实时数据处理能力, 以及在分布式和云数据库环境中优化数据同步和处理能力. 数据库的智能化将成为未来数据库发展的必然趋势, 智能数据库也会为日益增长的应用数据需求提供更加强大和有效的解决方案.

References

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

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

[3] Ding JL, Marcus R, Kipf A, Nathan V, Nrusimha A, Vaidya K, van Renen A, Kraska T. SageDB: An instance-optimized data analytics system. Proc. of the VLDB Endowment, 2022, 15(13): 4062–4078. [doi: 10.14778/3565838.3565857]

[4] Kraska T, Alizadeh M, Beutel A, Chi EH, Kristo A, Leclerc G, Madden S, Mao HZ, Nathan V. SageDB: A learned database system. 2019. https://www.cidrdb.org/cidr2019/papers/p117-kraska-cidr19.pdf

[5] Ma L, Zhang W, Jiao J, Wang WW, Butrovich M, Lim WS, Menon P, Pavlo A. MB2: Decomposed behavior modeling for self-driving database management systems. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 1248–1261. [doi: 10.1145/3448016.3457276]

[6] Li GL, Zhou XH, Sun J, Yu X, Han Y, Jin LY, Li WB, Wang TQ, Li SF. OpenGauss: An autonomous database system. Proc. of the VLDB Endowment, 2021, 14(12): 3028–3042. [doi: 10.14778/3476311.3476380]

[7] Zhao ZH, Cai SF, Gao HT, Pan HX, Xiang SQ, Xing NL, Chen G, Ooi BC, Shen YY, Wu YC, Zhang MH. NeurDB: On the design and implementation of an AI-powered autonomous database. arXiv:2408.03013, 2025.

[8] Helskyaho H, Yu J, Yu K. Machine Learning for Oracle Database Professionals: Deploying Model-driven Applications and Automation Pipelines. Berkeley: Apress, 2021. [doi: 10.1007/978-1-4842-7032-5]

[9] Barnes J. Microsoft Azure Essentials Azure Machine Learning. Washington: Microsoft Press, 2015.

[10] Narula S, Jain A, Prachi. Cloud computing security: Amazon Web service. In: Proc. of the 5th Int'l Conf. on Advanced Computing & Communication Technologies. Haryana: IEEE, 2015. 501–505. [doi: 10.1109/ACCT.2015.20]

[11] Verbitski A, Gupta A, Saha D, Brahmadesam M, Gupta K, Mittal R, Krishnamurthy S, Maurice S, Kharatishvili T, Bao XF. Amazon Aurora: Design considerations for high throughput cloud-native relational databases. In: Proc. of the 2017 ACM Int'l Conf. on Management of Data. Chicago: ACM, 2017. 1041–1052. [doi: 10.1145/3035918.3056101]

[12] 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. [doi: 10.1145/2094114.2094126]

[13] Gupta A, Tyagi S, Panwar N, Sachdeva S, Saxena U. NoSQL databases: Critical analysis and comparison. In: Proc. of the 2017 Int'l Conf. on Computing and Communication Technologies for Smart Nation. Gurgaon: IEEE, 2017. 293–299. [doi: 10.1109/IC3TSN.2017.8284494]

[14] Bhattacharyya A, Chakravarty D. Graph database: A survey. In: Proc. of the 2020 Int'l Conf. on Computer, Electrical & Communication Engineering (ICCECE). Kolkata: IEEE, 2020. 1–8. [doi: 10.1109/ICCECE48148.2020.9223105]

[15] Zhong V, Xiong CM, Socher R. Seq2SQL: Generating structured queries from natural language using reinforcement learning. arXiv:1709.00103, 2017.

[16] Xu XJ, Liu C, Song D. SQLNet: Generating structured queries from natural language without reinforcement learning. arXiv:1711.04436, 2017.

[17] Shi TZ, Tatwawadi K, Chakrabarti K, Mao Y, Polozov O, Chen WZ. IncSQL: Training incremental text-to-SQL parsers with non-deterministic oracles. arXiv:1809.05054, 2018.

[18] Yu T, Li ZF, Zhang ZL, Zhang R, Radev D. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In: Proc. of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics. New Orleans: Association for Computational Linguistics, 2018. 588–594. [doi: 10.18653/v1/N18-2093]

[19] Bogin B, Gardner M, Berant J. Representing schema structure with graph neural networks for text-to-SQL parsing. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: Association for Computational Linguistics, 2019. 4560–4565. [doi: 10.18653/v1/P19-1448]

[20] Bogin B, Gardner M, Berant J. Global reasoning over database structures for text-to-SQL parsing. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing. Hong Kong: Association for Computational Linguistics, 2019. 3659–3664. [doi: 10.18653/v1/D19-1378]

[21] Wang BL, Shin R, Liu XD, Polozov O, Richardson M. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020. 7567–7578. [doi: 10.18653/v1/2020.acl-main.677]

[22] Cao RS, Chen L, Chen Z, Zhao YB, Zhu S, Yu K. LGESQL: Line graph enhanced text-to-SQL model with mixed local and non-local relations. In: Proc. of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th Int'l Joint Conf. on Natural Language Processing. Association for Computational Linguistics, 2021. 2541–2555. [doi: 10.18653/v1/2021.acl-long.198]

[23] Cai RC, Yuan JJ, Xu BY, Hao ZF. SADGA: Structure-aware dual graph aggregation network for text-to-SQL. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Curran Associates Inc., 2021. 587.

[24] Hui BY, Geng RY, Wang LH, Qin BW, Li YY, Li BW, Sun J, Li YB. S^2SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers. In: Proc. of the 2022 Findings of the Association for Computational Linguistics. Dublin: Association for Computational Linguistics, 2022. 1254–1262. [doi: 10.18653/v1/2022.findings-acl.99]

[25] Chen Z, Chen L, Zhao YB, Cao RS, Xu ZH, Zhu S, Yu K. ShadowGNN: Graph projection neural network for text-to-SQL parser. arXiv:2104.04689, 2021.

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

[27] He PC, Mao Y, Chakrabarti K, Chen WZ. X-SQL: Reinforce schema representation with context. arXiv:1908.08113, 2019.

[28] Hwang W, Yim J, Park S, Seo M. A comprehensive exploration on WikiSQL with table-aware word contextualization. arXiv: 1902.01069, 2019.

[29] Xie TB, Wu CH, Shi P, Zhong RQ, Scholak T, Yasunaga M, Wu CS, Zhong M, Yin PC, Wang SI, Zhong V, Wang BL, Li CZ, Boyle C, Ni AS, Yao ZY, Radev D, Xiong CM, Kong LP, Zhang R, Smith NA, Zettlemoyer L, Yu T. UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In: Proc. of the 2022 Conf. on Empirical Methods in Natural Language Processing. Abu Dhabi: Association for Computational Linguistics, 2022. 602–631. [doi: 10.18653/v1/2022.emnlp-main.39]

[30] Scholak T, Li R, Bahdanau D, de Vries H, Pal C. DuoRAT: Towards simpler text-to-SQL models. In: Proc. of the 2021 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2021. 1313–1321. [doi: 10.18653/v1/2021.naacl-main.103]

[31] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional Transformers for language understanding. In: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long and Short Papers). Minneapolis: Association for Computational Linguistics, 2019. 4171–4186. [doi: 10.18653/v1/N19-1423]

[32] Brown TB, Mann B, Ryder N, *et al*. Language models are few-shot learners. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 159.

[33] Choi D, Shin MC, Kim E, Shin DR. RYANSQL: Recursively applying sketch-based slot fillings for complex text-to-SQL in cross-domain databases. Computational Linguistics, 2021, 47(2): 309–332. [doi: 10.1162/coli_a_00403]

[34] Guo JQ, Zhan ZC, Gao Y, Xiao Y, Lou JG, Liu T, Zhang DM. Towards complex text-to-SQL in cross-domain database with intermediate representation. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: Association for Computational Linguistics, 2019. 4524–4535. [doi: 10.18653/v1/P19-1444]

[35] Lyu Q, Chakrabarti K, Hathi S, Kundu S, Zhang JW, Chen Z. Hybrid ranking network for text-to-SQL. arXiv:2008.04759, 2020.

[36] Liu Q, Yang DJ, Zhang JH, Guo JQ, Zhou B, Lou JG. Awakening latent grounding from pretrained language models for semantic parsing. In: Proc. of the 2021 Findings of the Association for Computational Linguistics. Association for Computational Linguistics,

2021. 1174–1189. [doi: 10.18653/v1/2021.findings-acl.100]

[37] Yin PC, Neubig G, Yih WT, Riedel S. TaBERT: Pretraining for joint understanding of textual and tabular data. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020. 8413–8426. [doi: 10.18653/v1/2020.acl-main.745]

[38] Yu T, Wu CS, Lin XV, Wang BL, Tan YC, Yang XY, Radev DR, Socher R, Xiong CM. GraPPa: Grammar-augmented pre-training for table semantic parsing. In: Proc. of the 9th Int'l Conf. on Learning Representations. OpenReview.net, 2021.

[39] Shi P, Ng P, Wang ZG, Zhu HH, Li AH, Wang J, Nogueira dos Santos C, Xiang B. Learning contextual representations for semantic parsing with generation-augmented pre-training. In: Proc. of the 35th AAAI Conf. on Artificial Intelligence. AAAI, 2020. 13806–13814. [doi: 10.1609/aaai.v35i15.17627]

[40] Yan Z, Tang DY, Duan N, Liu SJ, Wang WD, Jiang DX, Zhou M, Li ZJ. Assertion-based QA with question-aware open information extraction. In: Proc. of the 32nd AAAI Conf. on Artificial Intelligence. New Orleans: AAAI, 2018. 6021–6028. [doi: 10.1609/aaai.v32i1.12052]

[41] Yu T, Yasunaga M, Yang K, Zhang R, Wang DX, Li ZF, Radev D. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In: Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing. Brussels: Association for Computational Linguistics, 2018. 1653–1663. [doi: 10.18653/v1/D18-1193]

[42] Rubin O, Berant J. SmBoP: Semi-autoregressive bottom-up semantic parsing. In: Proc. of the 2021 Conf. of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2021. 311–324. [doi: 10.18653/v1/2021.naacl-main.29]

[43] Dong L, Lapata M. Coarse-to-fine decoding for neural semantic parsing. In: Proc. of the 56th Annual Meeting of the Association for Computational Linguistics. Melbourne: Association for Computational Linguistics, 2018. 731–742. [doi: 10.18653/v1/P18-1068]

[44] Guo HL, Gao T, Wang F, Ma C. Bidirectional attention for SQL generation. In: Proc. of the 4th IEEE Int'l Conf. on Cloud Computing and Big Data Analysis. Chengdu: IEEE, 2019. 676–682. [doi: 10.1109/ICCCBDA.2019.8725626]

[45] Wang BL, Titov I, Lapata M. Learning semantic parsers from denotations with latent structured alignments and abstract programs. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing. Hong Kong: Association for Computational Linguistics, 2019. 3774–3785. [doi: 10.18653/v1/D19-1391]

[46] Brunner U, Stockinger K. ValueNet: A natural language-to-SQL system that learns from database information. In: Proc. of the 37th IEEE Int'l Conf. on Data Engineering. Chania: IEEE, 2021. 2177–2182. [doi: 10.1109/ICDE51399.2021.00220]

[47] Gan YJ, Chen XY, Xie JX, Purver M, Woodward JR, Drake J, Zhang QF. Natural SQL: Making SQL easier to infer from natural language specifications. arXiv:2109.05153, 2021.

[48] Suhr A, Chang MW, Shaw P, Lee K. Exploring unexplored generalization challenges for cross-database semantic parsing. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020. 8372–8388. [doi: 10.18653/v1/2020.acl-main.742]

[49] Yu T, Zhang R, Yang K, Yasunaga M, Wang DX, Li ZF, Ma J, Li I, Yao QN, Roman S, Zhang ZL, Radev D. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. arXiv:1809.08887, 2019.

[50] Liu AW, Hu XM, Wen LJ, Yu PS. A comprehensive evaluation of ChatGPT's zero-shot text-to-SQL capability. arXiv:2303.13547, 2023.

[51] Rajkumar N, Li R, Bahdanau D. Evaluating the text-to-SQL capabilities of large language models. arXiv:2204.00498, 2022.

[52] Chang SC, Fosler-Lussier E. Selective demonstrations for cross-domain text-to-SQL. In: Proc. of the 2023 Findings of the Association for Computational Linguistics. Singapore: Association for Computational Linguistics, 2023. 14174–14189. [doi: 10.18653/v1/2023.findings-emnlp.944]

[53] Pourreza M, Rafiei D. DTS-SQL: Decomposed text-to-SQL with small large language models. In: Proc. of the 2024 Findings of the Association for Computational Linguistics. Miami: Association for Computational Linguistics, 2024. 8212–8220. [doi: 10.18653/v1/2024.findings-emnlp.481]

[54] Pourreza M, Rafiei D. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In: Proc. of the 37th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2023. 1577.

[55] Li ZS, Wang X, Zhao JJ, Yang S, Du GQ, Hu XR, Zhang B, Ye YX, Li ZY, Zhao R, Mao HY. PET-SQL: A prompt-enhanced two-round refinement of text-to-SQL with cross-consistency. arXiv:2403.09732, 2024.

[56] Fan YK, He ZY, Ren TH, Huang C, Jing YN, Zhang K, Wang XS. Metasql: A generate-then-rank framework for natural language to SQL translation. In: Proc. of the 40th IEEE Int'l Conf. on Data Engineering. Utrecht: IEEE, 2024. 1765–1778. [doi: 10.1109/ICDE60146.2024.00143]

[57] Tai CY, Chen ZR, Zhang TS, Deng X, Sun H. Exploring chain-of-thought style prompting for text-to-SQL. arXiv:2305.14215, 2023.

[58] Zhang HC, Cao RS, Chen L, Xu HS, Yu K. ACT-SQL: In-context learning for text-to-SQL with automatically-generated chain-of-thought. In: Proc. of the 2023 Findings of the Association for Computational Linguistics. Singapore: Association for Computational Linguistics, 2023. 3501–3532. [doi: 10.18653/v1/2023.findings-emnlp.227]

[59] Sun RX, Arik SÖ, Muzio A, Miculicich L, Gundabathula S, Yin PC, Dai HJ, Nakhost H, Sinha R, Wang ZF, Pfister T. SQL-PaLM: Improved large language model adaptation for text-to-SQL (extended). arXiv:2306.00739, 2024.

[60] Shi FD, Fried D, Ghazvininejad M, Zettlemoyer L, Wang SI. Natural language to code translation with execution. In: Proc. of the 2022 Conf. on Empirical Methods in Natural Language Processing. Abu Dhabi: Association for Computational Linguistics, 2022. 3533–3546. [doi: 10.18653/v1/2022.emnlp-main.231]

[61] Dong XM, Zhang C, Ge YH, Mao YR, Gao YJ, Chen L, Lin JS, Lou DF. C3: Zero-shot text-to-SQL with ChatGPT. arXiv:2307.07306, 2023.

[62] Gao DW, Wang HB, Li YL, Sun XY, Qian YC, Ding BL, Zhou JR. Text-to-SQL empowered by large language models: A benchmark evaluation. Proc. of the VLDB Endowment, 2024, 17(5): 1132–1145. [doi: 10.14778/3641204.364122]

[63] Li HY, Zhang J, Liu HB, Fan J, Zhang XK, Zhu J, Wei RJ, Pan HY, Li CP, Chen H. CodeS: Towards building open-source language models for text-to-SQL. Proc. of the ACM on Management of Data, 2024, 2(3): 127. [doi: 10.1145/3654930]

[64] Liu YH, Han TL, Ma SY, Zhang JY, Yang YY, Tian JM, He H, Li AT, He MS, Liu ZL, Wu ZH, Zhao L, Zhu DJ, Li X, Qiang N, Shen DG, Liu TM, Ge B. Summary of ChatGPT-related research and perspective towards the future of large language models. Meta-Radiology, 2023, 1(2): 100017. [doi: 10.1016/j.metrad.2023.100017]

[65] Zha L, Zhou J, Li L, Wang R, Huang Q, Yang S, Yuan J, Su C, Li X, Su A, Zhang T, Zhou C, Shou K, Wang M, Zhu W, Lu G, Ye C, Ye Y, Ye W, Zhang Y, Deng X, Xu J, Wang H, Chen G, Zhao J. TableGPT: Towards unifying tables, nature language and commands into one GPT. arXiv:2307.08674, 2023.

[66] Deng Y, Lei WQ, Zhang WX, Lam W, Chua TS. PACIFIC: Towards proactive conversational question answering over tabular and textual data in finance. In: Proc. of the 2022 Conf. on Empirical Methods in Natural Language Processing. Abu Dhabi: Association for Computational Linguistics, 2022. 6970–6984. [doi: 10.18653/v1/2022.emnlp-main.469]

[67] Zhao YL, Long YT, Liu HJ, Kamoi R, Nan LY, Chen LH, Liu YX, Tang XR, Zhang R, Cohan A. DocMath-eval: Evaluating math reasoning capabilities of LLMs in understanding long and specialized documents. In: Proc. of the 62nd Annual Meeting of the Association for Computational Linguistics. Bangkok: Association for Computational Linguistics, 2024. 16103–16120. [doi: 10.18653/v1/2024.acl-long.852]

[68] Sui Y, Zou JR, Zhou MY, He XY, Du L, Han S, Zhang DM. TAP4LLM: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning. In: Proc. of the 2024 Findings of the Association for Computational Linguistics. Miami: Association for Computational Linguistics, 2024. 10306–10323. [doi: 10.18653/v1/2024.findings-emnlp.603]

[69] Sundar AS, Heck L. cTBLS: Augmenting large language models with conversational tables. In: Proc. of the 5th Workshop on NLP for Conversational AI. Toronto: Association for Computational Linguistics, 2023. 59–70. [doi: 10.18653/v1/2023.nlp4convai-1.6]

[70] Ye JJ, Chen XT, Xu N, Zu C, Shao ZK, Liu SC, Cui YH, Zhou ZY, Gong C, Shen Y, Zhou J, Chen SM, Gui T, Zhang Q, Huang XJ. A comprehensive capability analysis of GPT-3 and GPT-3.5 series models. arXiv:2303.10420, 2023.

[71] Sui Y, Zhou MY, Zhou MJ, Han S, Zhang DM. Table meets LLM: Can large language models understand structured table data? A benchmark and empirical study. In: Proc. of the 17th ACM Int'l Conf. on Web Search and Data Mining. Mexico: ACM, 2024. 645–654. [doi: 10.1145/3616855.3635752]

[72] Yu T, Zhang R, Yasunaga M, Tan YC, Lin XV, Li SY, Er HY, Li I, Pang B, Chen T, Ji E, Dixit S, Proctor D, Shim S, Kraft J, Zhang V, Xiong CM, Socher R, Radev D. SParC: Cross-domain semantic parsing in context. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: Association for Computational Linguistics, 2019. 4511–4523. [doi: 10.18653/v1/P19-1443]

[73] Weng LX, Tang YH, Feng YCJ, Chang Z, Chen RQ, Feng HZ, Hou C, Huang DQ, Li Y, Rao HN, Wang HM, Wei CS, Yang XF, Zhang YH, Zheng YF, Huang XQ, Zhu MF, Ma YX, Cui B, Chen P, Chen W. DataLab: A unified platform for LLM-powered business intelligence. In: Proc. of the 41st IEEE Int'l Conf. on Data Engineering. Hong Kong: IEEE, 2025. 4346–4359. [doi: 10.1109/ICDE65448.2025.00326]

[74] Katsogiannis-Meimarakis G, Koutrika G. A survey on deep learning approaches for text-to-SQL. The VLDB Journal, 2023, 32(4): 905–936. [doi: 10.1007/s00778-022-00776-8]

[75] Deng NH, Chen YL, Zhang Y. Recent advances in text-to-SQL: A survey of what we have and what we expect. In: Proc. of the 29th Int'l Conf. on Computational Linguistics. Gyeongju: Int'l Committee on Computational Linguistics, 2022. 2166–2187.

[76] Hong ZJ, Yuan Z, Zhang QG, Chen H, Dong JN, Huang FR, Huang X. Next-generation database interfaces: A survey of LLM-based text-to-SQL. arXiv:2406.08426, 2025.

[77] Fang X, Xu WJ, Tan FA, Zhang JQ, Hu ZN, Qi YJ, Nickleach S, Socolinsky D, Sengamedu S, Faloutsos C. Large language models (LLMs) on tabular data: Prediction, generation, and understanding—A survey. arXiv:2402.17944, 2024.

[78] Chen HF, Zhang WX, Jiang GF. Experience transfer for the configuration tuning in large-scale computing systems. IEEE Trans. on Knowledge and Data Engineering, 2011, 23(3): 388–401. [doi: 10.1109/TKDE.2010.121]

[79] Zhu YQ, Liu JX, Guo MY, Bao YG, Ma WL, Liu ZY, Song KP, Yang YC. BestConfig: Tapping the performance potential of systems via automatic configuration tuning. In: Proc. of the 2017 Symp. on Cloud Computing. Santa Clara: ACM, 2017. 338–350. [doi: 10.1145/3127479.3128605]

[80] Duan SY, Thummala V, Babu S. Tuning database configuration parameters with iTuned. Proc. of the VLDB Endowment, 2009, 2(1): 1246–1257. [doi: 10.14778/1687627.1687767]

[81] van Aken D, Pavlo A, Gordon GJ, Zhang BH. Automatic database management system tuning through large-scale machine learning. In: Proc. of the 2017 ACM Int'l Conf. on Management of Data. Chicago: ACM, 2017. 1009–1024. [doi: 10.1145/3035918.3064029]

[82] Fekry A, Carata L, Pasquier T, Rice A, Hopper A. Tuneful: An online significance-aware configuration tuner for big data analytics. arXiv:2001.08002, 2020.

[83] Zhang XY, Wu H, Chang Z, Jin SW, Tan J, Li FF, Zhang TY, Cui B. ResTune: Resource oriented tuning boosted by meta-learning for cloud databases. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 2102–2114. [doi: 10.1145/3448016.3457291]

[84] Cereda S, Valladares S, Cremonesi P, Doni S. CGPTuner: A contextual Gaussian process bandit approach for the automatic tuning of it configurations under varying workload conditions. Proc. of the VLDB Endowment, 2021, 14(8): 1401–1413. [doi: 10.14778/3457390.3457404]

[85] Zhang XY, Wu H, Li Y, Tan J, Li FF, Cui B. Towards dynamic and safe configuration tuning for cloud databases. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 631–645. [doi: 10.1145/3514221.3526176]

[86] Kanellis K, Ding C, Kroth B, Müller A, Curino C, Venkataraman S. LlamaTune: Sample-efficient DBMS configuration tuning. Proc. of the VLDB Endowment, 2022, 15(11): 2953–2965. [doi: 10.14778/3551793.3551844]

[87] Wang QL, Wang RZ, Hu YX, Shi XH, Liu Z, Ma T, Song HB, Shi HY. KeenTune: Automated tuning tool for cloud application performance testing and optimization. In: Proc. of the 32nd ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Seattle: ACM, 2023. 1487–1490. [doi: 10.1145/3597926.3604920]

[88] Tan J, Zhang TY, Li FF, Chen J, Zheng QX, Zhang P, Qiao HL, Shi Y, Cao W, Zhang R. IBTune: Individualized buffer tuning for large-scale cloud databases. Proc. of the VLDB Endowment, 2019, 12(10): 1221–1234. [doi: 10.14778/3339490.3339503]

[89] Lin C, Zhuang JQ, Feng JD, Li H, Zhou XH, Li GL. Adaptive code learning for Spark configuration tuning. In: Proc. of the 38th IEEE Int'l Conf. on Data Engineering. Kuala Lumpur: IEEE, 2022. 1995–2007. [doi: 10.1109/ICDE53745.2022.00195]

[90] Zhang J, Liu Y, Zhou K, Li GL, Xiao ZL, Cheng B, Xing JS, Wang YT, Cheng TH, Liu L, Ran MW, Li ZK. An end-to-end automatic cloud database tuning system using deep reinforcement learning. In: Proc. of the 2019 Int'l Conf. on Management of Data. Amsterdam: ACM, 2019. 415–432. [doi: 10.1145/3299869.3300085]

[91] Li GL, Zhou XH, Li SF, Gao B. QTune: A query-aware database tuning system with deep reinforcement learning. Proc. of the VLDB Endowment, 2019, 12(12): 2118–2130. [doi: 10.14778/3352063.3352129]

[92] Trummer I. DB-BERT: A database tuning tool that "reads the manual". In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 190–203. [doi: 10.1145/3514221.3517843]

[93] Cai BQ, Liu Y, Zhang C, Zhang GY, Zhou K, Liu L, Li CH, Cheng B, Yang J, Xing JS. HUNTER: An online cloud database hybrid tuning system for personalized requirements. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 646–659. [doi: 10.1145/3514221.3517882]

[94] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. In: Proc. of the 4th Int'l Conf. on Learning Representations. San Juan: ICLR, 2016.

[95] Lyu JF, Ma XT, Yan JP, Li X. Efficient continuous control with double actors and regularized critics. In: Proc. of the 36th AAAI Conf. on Artificial Intelligence. AAAI, 2022. 7655–7663. [doi: 10.1609/aaai.v36i7.20732]

[96] Shlens J. A tutorial on principal component analysis. arXiv:1404.1100, 2014.

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

[98] Chaudhuri S, Narasayya V. AutoAdmin "what-if" index analysis utility. In: Proc. of the 1998 ACM SIGMOD Int'l Conf. on

Management of Data. Seattle: ACM, 1998. 367–378. [doi: 10.1145/276304.276337]

[99] Chaudhuri S, Datar M, Narasayya V. Index selection for databases: A hardness study and a principled heuristic solution. IEEE Trans. on Knowledge and Data Engineering, 2004, 16(11): 1313–1323. [doi: 10.1109/TKDE.2004.75]

[100] Schlosser R, Kossmann J, Boissier M. Efficient scalable multi-attribute index selection using recursive strategies. In: Proc of the 35th IEEE Int'l Conf. on Data Engineering. Macao: IEEE, 2019. 1238–1249. [doi: 10.1109/ICDE.2019.00113]

[101] Dash D, Polyzotis N, Ailamaki A. CoPhy: A scalable, portable, and interactive index advisor for large workloads. Proc. of the VLDB Endowment, 2011, 4(6): 362–372. [doi: 10.14778/1978665.1978668]

[102] Siddiqui T, Wu WT, Narasayya V, Chaudhuri S. DISTILL: Low-overhead data-driven techniques for filtering and costing indexes for scalable index tuning. Proc. of the VLDB Endowment, 2022, 15(10): 2019–2031. [doi: 10.14778/3547305.3547309]

[103] Siddiqui T, Jo S, Wu WT, Wang C, Narasayya V, Chaudhuri S. ISUM: Efficiently compressing large and complex workloads for scalable index tuning. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 660–673. [doi: 10.1145/3514221.3526152]

[104] Wu WT, Wang C, Siddiqui T, Wang JX, Narasayya V, Chaudhuri S, Bernstein PA. Budget-aware index tuning with reinforcement learning. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 1528–1541. [doi: 10.1145/3514221.3526128]

[105] Sharma A, Schuhknecht FM, Dittrich J. The case for automatic database administration using deep reinforcement learning. arXiv:1801.05643, 2018.

[106] Lan H, Bao ZF, Peng YW. An index advisor using deep reinforcement learning. In: Proc. of the 29th ACM Int'l Conf. on Information & Knowledge Management. ACM, 2020. 2105–2108. [doi: 10.1145/3340531.3412106]

[107] Schnaitter K, Abiteboul S, Milo T, Polyzotis N. On-line index selection for shifting workloads. In: Proc. of the 23rd IEEE Int'l Conf. on Data Engineering Workshop. Istanbul: IEEE, 2007. 459–468. [doi: 10.1109/ICDEW.2007.4401029]

[108] Ma L, van Aken D, Hefny A, Mezerhane G, Pavlo A, Gordon GJ. Query-based workload forecasting for self-driving database management systems. In: Proc. of the 2018 Int'l Conf. on Management of Data. Houston: ACM, 2018. 631–645. [doi: 10.1145/3183713.3196908]

[109] Bruno N, Chaudhuri S. To tune or not to tune?: A lightweight physical design alerter. In: Proc. of the 32nd Int'l Conf. on Very Large Data Bases. Seoul: VLDB Endowment, 2006. 499–510.

[110] Bruno N, Chaudhuri S. An online approach to physical design tuning. In: Proc. of the 23rd IEEE Int'l Conf. on Data Engineering. Istanbul: IEEE, 2007. 826–835. [doi: 10.1109/ICDE.2007.367928]

[111] Schnaitter K, Polyzotis N. Semi-automatic index tuning: Keeping DBAs in the loop. Proc. of the VLDB Endowment, 2012, 5(5): 478–489. [doi: 10.14778/2140436.2140444]

[112] Yadav R, Valluri SR, Zaït M. AIM: A practical approach to automated index management for SQL databases. In: Proc. of the 39th IEEE Int'l Conf. on Data Engineering. Anaheim: IEEE, 2023. 3349–3362. [doi: 10.1109/ICDE55515.2023.00257]

[113] Shi JC, Cong G, Li XL. Learned index benefits: Machine learning based index performance estimation. Proc. of the VLDB Endowment, 2022, 15(13): 3950–3962. [doi: 10.14778/3565838.3565848]

[114] Perera RM, Oetomo B, Rubinstein BIP, Borovica-Gajic R. DBA Bandits: Self-driving index tuning under ad-hoc, analytical workloads with safety guarantees. In: Proc. of the 37th IEEE Int'l Conf. on Data Engineering. Chania: IEEE, 2021. 600–611. [doi: 10.1109/ICDE51399.2021.00058]

[115] Kossmann J, Kastius A, Schlosser R. SWIRL: Selection of workload-aware indexes using reinforcement learning. In: Proc. of the 25th Int'l Conf. on Extending Database Technology. Edinburgh: OpenProceedings.org, 2022. 2:155–2:168.

[116] Sadri Z, Gruenwald L, Lead E. DRLindex: Deep reinforcement learning index advisor for a cluster database. In: Proc. of the 24th Symp. on Int'l Database Engineering & Applications. Seoul: ACM, 2020. 11. [doi: 10.1145/3410566.3410603]

[117] Perera RM, Oetomo B, Rubinstein BIP, Borovica-Gajic R. HMAB: Self-driving hierarchy of bandits for integrated physical database design tuning. Proc. of the VLDB Endowment, 2022, 16(2): 216–229. [doi: 10.14778/3565816.3565824]

[118] Zhou XH, Liu LY, Li WB, Jin LY, Li SF, Wang TQ, Feng JH. AutoIndex: An incremental index management system for dynamic workloads. In: Proc. of the 38th IEEE Int'l Conf. on Data Engineering. Kuala Lumpur: IEEE, 2022. 2196–2208. [DOI: 10.1109/ICDE53745.2022.00210]

[119] Chaudhuri S, Narasayya V. Anytime algorithm of database tuning advisor for Microsoft SQL server. 2020. https://www.microsoft.com/en-us/research/wp-content/uploads/2020/06/Anytime-Algorithm-of-Database-Tuning-Advisor-for-Microsoft-SQL-Server.pdf

[120] Valentin G, Zuliani M, Zilio DC, Lohman G, Skelley A. DB2 Advisor: An optimizer smart enough to recommend its own indexes. In: Proc. of the 16th Int'l Conf. on Data Engineering. San Diego: IEEE, 2000. 101–110. [doi: 10.1109/ICDE.2000.839397]

[121] Bruno N, Chaudhuri S. Automatic physical database tuning: A relaxation-based approach. In: Proc. of the 2005 ACM SIGMOD Int'l Conf. on Management of Data. Baltimore: ACM, 2005. 227–238. [doi: 10.1145/1066157.1066184]

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

[123] Hammer M, Chan A. Index selection in a self-adaptive data base management system. In: Proc. of the 1976 ACM SIGMOD Int'l Conf. on Management of Data. Washington: ACM, 1976. 1–8. [doi: 10.1145/509383.509385]

[124] Liu XZ, Yin Z, Zhao C, Ge CC, Chen L, Gao YJ, Li DM, Wang ZT, Liang GZ, Tan J, Li FF. PinSQL: Pinpoint root cause SQLs to resolve performance issues in cloud databases. In: Proc. of the 38th IEEE Int'l Conf. on Data Engineering. Kuala Lumpur: IEEE, 2022. 2549–2561. [doi: 10.1109/ICDE53745.2022.00236]

[125] Laptev N, Amizadeh S, Flint I. Generic and scalable framework for automated time-series anomaly detection. In: Proc. of the 21st ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Sydney: ACM, 2015. 1939–1947. [doi: 10.1145/2783258.2788611]

[126] Siffer A, Fouque PA, Termier A, Largouët C. Anomaly detection in streams with extreme value theory. In: Proc. of the 23rd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Halifax: ACM, 2017. 1067–1075. [doi: 10.1145/3097983.3098144]

[127] Ma MH, Yin Z, Zhang SL, Wang S, Zheng C, Jiang XH, Hu HW, Luo C, Li YL, Qiu NJ, Li FF, Chen CC, Pei D. Diagnosing root causes of intermittent slow queries in cloud databases. Proc. of the VLDB Endowment, 2020, 13(8): 1176–1189. [doi: 10.14778/3389133.3389136]

[128] Liu P, Zhang SL, Sun YQ, Meng Y, Yang JH, Pei D. FluxInfer: Automatic diagnosis of performance anomaly for online database system. In: Proc. of the 39th IEEE Int'l Performance Computing and Communications Conf. Austin: IEEE, 2020. 1–8. [doi: 10.1109/IPCCC50635.2020.9391550]

[129] Yoon DY, Niu N, Mozafari B. DBSherlock: A performance diagnostic tool for transactional databases. In: Proc. of the 2016 Int'l Conf. on Management of Data. San Francisco: ACM, 2016. 1599–1614. [doi: 10.1145/2882903.2915218]

[130] Jeyakumar V, Madani O, Parandeh A, Kulshreshtha A, Zeng WF, Yadav N. ExplainIt!—A declarative root-cause analysis engine for time series data. In: Proc. of the 2019 Int'l Conf. on Management of Data. Amsterdam: ACM, 2019. 333–348. [doi: 10.1145/3299869.3314048]

[131] Jin LY, Li GL. AI-based database performance diagnosis. Ruan Jian Xue Bao/Journal of Software, 2021, 32(3): 845–858 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6177.htm [doi: 10.13328/j.cnki.jos.006177]

[132] Dintyala P, Narechania A, Arulraj J. SQLCheck: Automated detection and diagnosis of SQL anti-patterns. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 2331–2345. [doi: 10.1145/3318464.3389754]

[133] Sharma T, Fragkoulis M, Rizou S, Bruntink M, Spinellis D. Smelly relations: Measuring and understanding database schema quality. In: Proc. of the 40th IEEE/ACM Int'l Conf. on Software Engineering: Software Engineering in Practice Track. Gothenburg: IEEE, 2018. 55–64.

[134] Mao YT, Yuan S, Cui N, Du TJ, Shen BJ, Chen YT. DeFiHap: Detecting and fixing HiveQL anti-patterns. Proc. of the VLDB Endowment, 2021, 14(12): 2671–2674. [doi: 10.14778/3476311.3476316]

[135] Khoussainova N, Balazinska M, Suciu D. PerfXplain: Debugging mapreduce job performance. Proc. of the VLDB Endowment, 2012, 5(7): 598–609. [doi: 10.14778/2180912.2180913]

[136] Das S, Li F, Narasayya VR, König AC. Automated demand-driven resource scaling in relational database-as-a-service. In: Proc. of the 2016 Int'l Conf. on Management of Data. San Francisco: ACM, 2016. 1923–1934. [doi: 10.1145/2882903.2903733]

[137] Higginson AS, Dediu M, Arsene O, Paton NW, Embury SM. Database workload capacity planning using time series analysis and machine learning. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 769–783. [doi: 10.1145/3318464.3386140]

[138] Poppe O, Amuneke T, Banda D, *et al*. Seagull: An infrastructure for load prediction and optimized resource allocation. Proc. of the VLDB Endowment, 2020, 14(2): 154–162. [doi: 10.14778/3425879.3425886]

[139] Pimpley A, Li S, Sen R, Srinivasan S, Jindal A. Towards optimal resource allocation for big data analytics. In: Proc. of the 25th Int'l Conf. on Extending Database Technology. Edinburgh: OpenProceedings.org, 2022. 2: 338–2: 350.

[140] König AC, Shan Y, Ziegler T, Kakaraparthy A, Lang W, Moeller J, Kalhan A, Narasayya V. Tenant placement in over-subscribed database-as-a-service clusters. Proc. of the VLDB Endowment, 2022, 15(11): 2559–2571. [doi: 10.14778/3551793.3551814]

[141] Sun LM, Gong SJ, Zhang TY, Jiang FX, Zhao ZB, Chen JJ, Zhang XY. SUFS: A generic storage usage forecasting service through adaptive ensemble learning. In: Proc. of the 39th IEEE Int'l Conf. on Data Engineering. Anaheim: IEEE, 2023. 3168–3181. [doi: 10.1109/ICDE55515.2023.00243]

[142] Saxena G, Rahman M, Chainani N, Lin CB, Caragea G, Chowdhury F, Marcus R, Kraska T, Pandis I, Narayanaswamy B. Auto-WLM: Machine learning enhanced workload management in Amazon Redshift. In: Proc. of Companion of the 2023 Int'l Conf. on Management of Data. Seattle: ACM, 2023. 225–237. [doi: 10.1145/3555041.3589677]

[143] Zhang LX, Chai CL, Zhou XH, Li GL. LearnedSQLGen: Constraint-aware SQL generation using reinforcement learning. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 945–958. [doi: 10.1145/3514221.3526155]

[144] Sun Z, Cheung A. Query aware synthetic data generation. Technical Report, UCB/EECS-2023-124, Berkeley: University of California, 2023.

[145] Weng SY, Wang QS, Qu LY, Zhang R, Cai P, Qian WN, Zhou AY. Lauca: A workload duplicator for benchmarking transactional database performance. IEEE Trans. on Knowledge and Data Engineering, 2024, 36(7): 3180–3194. [doi: 10.1109/TKDE.2024.3360116]

[146] Qu LY, Li YM, Zhang R, Chen T, Shu K, Qian WN, Zhou AY. Application-oriented workload generation for transactional database performance evaluation. In: Proc. of the 38th IEEE Int'l Conf. on Data Engineering. Kuala Lumpur: IEEE, 2022. 420–432. [doi: 10.1109/ICDE53745.2022.00036]

[147] Lerner A, Jasny M, Jepsen T, Binnig C, Cudré-Mauroux P. DBMS annihilator: A high-performance database workload generator in action. Proc. of the VLDB Endowment, 2022, 15(12): 3682–3685. [doi: 10.14778/3554821.3554874]

[148] Holze M, Gaidies C, Ritter N. Consistent on-line classification of DBs workload events. In: Proc. of the 18th ACM Conf. on Information and Knowledge Management. Hong Kong: ACM, 2009. 1641–1644. [doi: 10.1145/1645953.1646193]

[149] Holze M, Ritter N. Towards workload shift detection and prediction for autonomic databases. In: Proc of the 1st ACM Ph.D. Workshop in CIKM. Lisbon: ACM, 2007. 109–116. [doi: 10.1145/1316874.1316892]

[150] Wu PZ, Ives ZG. Modeling shifting workloads for learned database systems. Proc. of the ACM on Management of Data, 2024, 2(1): 38. [doi: 10.1145/3639293]

[151] Gao YN, Huang XQ, Zhou XH, Gao XF, Li GL, Chen GH. DBAugur: An adversarial-based trend forecasting system for diversified workloads. In: Proc. of the 39th IEEE Int'l Conf. on Data Engineering. Anaheim: IEEE, 2023. 27–39. [doi: 10.1109/ICDE55515.2023.00385]

[152] Wang JQ, Li TY, Wang AN, Liu XZ, Chen L, Chen J, Liu JY, Wu JY, Li FF, Gao YJ. Real-time workload pattern analysis for large-scale cloud databases. Proc. of the VLDB Endowment, 2023, 16(12): 3689–3701. [doi: 10.14778/3611540.3611557]

[153] Zhao XY, Zhou XH, Li GL. Automatic database knob tuning: A survey. IEEE Trans. on Knowledge and Data Engineering, 2023, 35(12): 12470–12490. [doi: 10.1109/TKDE.2023.3266893]

[154] Wang JH, Liu LQ, Li CX. A survey of database knob tuning with machine learning. In: Proc. of the 2nd IEEE Int'l Conf. on Electrical Engineering, Big Data and Algorithms. Changchun: IEEE, 2023. 1545–1550. [doi: 10.1109/EEBDA56825.2023.10090597]

[155] Li YY, Tian JK, Pu Z, Li CP, Chen H. Research survey on intelligent tuning of database knobs configuration. Chinese Journal of Computers, 2024, 47(8): 1901–1921 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2024.01901]

[156] Wu Y, Zhou XH, Zhang Y, Li GL. Automatic index tuning: A survey. IEEE Trans. on Knowledge and Data Engineering, 2024, 36(12): 7657–7676. [doi: 10.1109/TKDE.2024.3422006]

[157] Zhou W, Lin C, Zhou XH, Li GL. Breaking it down: An in-depth study of index advisors. Proc. of the VLDB Endowment, 2024, 17(10): 2405–2418. [doi: 10.14778/3675034.3675035]

[158] Comer D. Ubiquitous B-tree. ACM Computing Surveys (CSUR), 1979, 11(2): 121–137. [doi: 10.1145/356770.356776]

[159] Kraska T, Beutel A, Chi EH, Dean J, Polyzotis N. The case for learned index structures. In: Proc. of the 2018 Int'l Conf. on Management of Data. Houston: ACM, 2018. 489–504. [doi: 10.1145/3183713.3196909]

[160] Tang CZ, Wang YY, Dong ZY, Hu GS, Wang ZG, Wang MJ, Chen HB. XIndex: A scalable learned index for multicore data storage. In: Proc. of the 25th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming. San Diego: ACM, 2020. 308–320. [doi: 10.1145/3332466.3374547]

[161] Li PF, Hua Y, Jia JN, Zuo PF. FINEdex: A fine-grained learned index scheme for scalable and concurrent memory systems. Proc. of the VLDB Endowment, 2021, 15(2): 321–334. [doi: 10.14778/3489496.3489512]

[162] Wang YY, Tang CZ, Wang ZG, Chen HB. SIndex: A scalable learned index for string keys. In: Proc. of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems. Tsukuba: ACM, 2020. 17–24. [doi: 10.1145/3409963.3410496]

[163] Ding JL, Minhas UF, Yu J, Wang C, Do J, Li YN, Zhang HT, Chandramouli B, Gehrke J, Kossmann D, Lomet D, Kraska T. ALEX: An updatable adaptive learned index. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 969–984. [doi: 10.1145/3318464.3389711]

[164] Hadian A, Heinis T. MADEX: Learning-augmented algorithmic index structures. In: Proc. of the 2nd Int'l Workshop on Applied AI for Database Systems and Applications. Tokyo, 2020.

[165] Wu JC, Zhang Y, Chen SM, Wang J, Chen Y, Xing CX. Updatable learned index with precise positions. Proc. of the VLDB Endowment, 2021, 14(8): 1276–1288. [doi: 10.14778/3457390.3457393]

[166] Wang HX, Fu XY, Xu JL, Lu H. Learned index for spatial queries. In: Proc. of the 20th IEEE Int'l Conf. on Mobile Data Management. Hong Kong: IEEE, 2019. 569–574. [doi: 10.1109/MDM.2019.00121]

[167] Davitkova A, Milchevski E, Michel S. The ML-index: A multidimensional, learned index for point, range, and nearest-neighbor queries. In: Proc. of the 23rd Int'l Conf. on Extending Database Technology. Copenhagen: OpenProceedings.org, 2020. 407–410.

[168] Li PF, Lu H, Zheng Q, Yang L, Pan G. LISA: A learned index structure for spatial data. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 2119–2133. [doi: 10.1145/3318464.3389703]

[169] Hadian A, Kumar A, Heinis T. Hands-off model integration in spatial index structures. In: Proc. of the 2nd Int'l Workshop on Applied AI for Database Systems and Applications. Tokyo: AIDB@VLDB, 2020.

[170] Qi JZ, Liu GL, Jensen CS, Kulik L. Effectively learning spatial indices. Proc. of the VLDB Endowment, 2020, 13(12): 2341–2354. [doi: 10.14778/3407790.3407829]

[171] Yang ZH, Chandramouli B, Wang C, Gehrke J, Li YN, Minhas UF, Larson PÅ, Kossmann D, Acharya R. QD-Tree: Learning data layouts for big data analytics. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 193–208. [doi: 10.1145/3318464.3389770]

[172] Nathan V, Ding JL, Alizadeh M, Kraska T. Learning multi-dimensional indexes. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 985–1000. [doi: 10.1145/3318464.3380579]

[173] Ding JL, Nathan V, Alizadeh M, Kraska T. Tsunami: A learned multi-dimensional index for correlated data and skewed workloads. Proc. of the VLDB Endowment, 2020, 14(2): 74–86. [doi: 10.14778/3425879.3425880]

[174] Jagadish HV, Ooi BC, Tan KL, Yu C, Zhang R. IDistance: An adaptive B^+-tree based indexing method for nearest neighbor search. ACM Trans. on Database Systems (TODS), 2005, 30(2): 364–397. [doi: 10.1145/1071610.1071612]

[175] Pai SG, Mathioudakis M, Wang YH. Towards an instance-optimal Z-index. In: Proc. of the 4th Int'l Workshop on Applied AI for Database Systems and Applications. Sydney, 2022.

[176] Gao J, Cao X, Yao X, Zhang G, Wang W. LMSFC: A novel multidimensional index based on learned monotonic space filling curves. Proc. of the VLDB Endowment, 2023, 16(10): 2605–2617. [doi: 10.14778/3603581.3603598]

[177] Pai S, Mathioudakis M, Wang YH. WaZI: A learned and workload-aware Z-index. In: Proc. of the 27th Int'l Conf. on Extending Database Technology. Paestum: OpenProceedings.org, 2024. 559–571.

[178] Guttman A. R-trees: A dynamic index structure for spatial searching. ACM SIGMOD Record, 1984, 14(2): 47–57. [doi: 10.1145/971697.602266]

[179] Bentley JL. Multidimensional binary search trees used for associative searching. Communications of the ACM, 1975, 18(9): 509–517. [doi: 10.1145/361002.361007]

[180] Li Z, Chan TN, Yiu ML, Jensen CS. PolyFit: Polynomial-based indexing approach for fast approximate range aggregate queries. In: Proc. of the 24th Int'l Conf. on Extending Database Technology. Nicosia: OpenProceedings.org, 2021. 241–252.

[181] Peng YX, Zhou W, Zhang L, Du HL. A study of learned KD tree based on learned index. In: Proc. of the 2020 Int'l Conf. on Networking and Network Applications. Haikou: IEEE, 2020. 355–360. [doi: 10.1109/NaNA51271.2020.00067]

[182] Zhang SN, Ray S, Lu RX, Zheng YD. Efficient learned spatial index with interpolation function based learned model. IEEE Trans. on Big Data, 2023, 9(2): 733–745. [doi: 10.1109/TBDATA.2022.3186857]

[183] Hadian A, Ghaffari B, Wang TY, Heinis T. COAX: Correlation-aware indexing. In: Proc. of the 39th IEEE Int'l Conf. on Data Engineering Workshops. Anaheim: IEEE, 2023. 55–59. [doi: 10.1109/ICDEW58674.2023.00014]

[184] Lu Y, Shanbhag A, Jindal A, Madden S. AdaptDB: Adaptive partitioning for distributed joins. Proc. of the VLDB Endowment, 2017, 10(5): 589–600. [doi: 10.14778/3055540.3055551]

[185] Sun LW, Franklin MJ, Krishnan S, Xin RS. Fine-grained partitioning for aggressive data skipping. In: Proc. of the 2014 ACM SIGMOD Int'l Conf. on Management of Data. Snowbird Utah: ACM, 2014. 1115–1126. [doi: 10.1145/2588555.2610515]

[186] Parchas P, Naamad Y, van Bouwel P, Faloutsos C, Petropoulos M. Fast and effective distribution-key recommendation for amazon redshift. Proc. of the VLDB Endowment, 2020, 13(12): 2411–2423. [doi: 10.14778/3407790.3407834]

[187] Serafini M, Taft R, Elmore AJ, Pavlo A, Aboulnaga A, Stonebraker M. Clay: Fine-grained adaptive partitioning for general database schemas. Proc. of the VLDB Endowment, 2016, 10(4): 445–456. [doi: 10.14778/3025111.3025125]

[188] Eldeeb T, Chen ZN, Cidon A, Yang JF. Neuroshard: Towards automatic multi-objective sharding with deep reinforcement learning. In: Proc. of the 5th Int'l Workshop on Exploiting Artificial Intelligence Techniques for Data Management. Philadelphia: ACM, 2022. 1. [doi: 10.1145/3533702.3534908]

[189] Hilprecht B, Binnig C, Röhm U. Learning a partitioning advisor for cloud databases. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 143–157. [doi: 10.1145/3318464.3389704]

[190] 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. In: Proc. of the 1st Int'l Workshop on Exploiting Artificial Intelligence Techniques for Data Management. Houston: ACM, 2018. 1. [doi: 10.1145/3211954.3211956]

[191] Sharify S, Lu AL, Chen J, Bhattacharyya A, Hashemi A, Koudas N, Amza C. An improved dynamic vertical partitioning technique for semi-structured data. In: Proc. of the 2019 IEEE Int'l Symp. on Performance Analysis of Systems and Software. Madison: IEEE, 2019. 243–256. [doi: 10.1109/ISPASS.2019.00037]

[192] Grund M, Krüger J, Plattner H, Zeier A, Cudre-Mauroux P, Madden S. HYRISE: A main memory hybrid storage engine. Proc. of the VLDB Endowment, 2010, 4(2): 105–116. [doi: 10.14778/1921071.1921077]

[193] Alagiannis I, Idreos S, Ailamaki A. H2O: A hands-free adaptive store. In: Proc. of the 2014 ACM SIGMOD Int'l Conf. on Management of Data. Snowbird Utah: ACM, 2014. 1103–1114. [doi: 10.1145/2588555.2610502]

[194] Kang DH, Jiang RC, Blanas S. Jigsaw: A data storage and query processing engine for irregular table partitioning. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 898–911. [doi: 10.1145/3448016.3457547]

[195] Zapridou E, Mytilinis I, Ailamaki A. Dalton: Learned partitioning for distributed data streams. Proc. of the VLDB Endowment, 2022, 16(3): 491–504. [doi: 10.14778/3570690.3570699]

[196] Zhou XH, Li GL, Feng JH, Liu LY, Guo W. Grep: A graph learning based database partitioning system. Proc. of the ACM on Management of Data, 2023, 1(1): 94. [doi: 10.1145/3588948]

[197] Athanassoulis M, Bøgh KS, Idreos S. Optimal column layout for hybrid workloads. Proc. of the VLDB Endowment, 2019, 12(13): 2393–2407. [doi: 10.14778/3358701.3358707]

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

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

[200] Zhou XH, Li GL, Wu JM, Liu JS, Sun ZY, Zhang XN. A learned query rewrite system. Proc. of the VLDB Endowment, 2023, 16(12): 4110–4113. [doi: 10.14778/3611540.3611633]

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

[202] Zhu R, Wu ZN, Han YX, Zeng K, Pfadler A, Qian ZP, Zhou JR, Cui B. FLAT: Fast, lightweight and accurate method for cardinality estimation. Proc. of the VLDB Endowment, 2021, 14(9): 148–1502. [doi: 10.14778/3461535.3461539]

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

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

[205] Germain M, Gregor K, Murray I, Larochelle H. MADE: Masked autoencoder for distribution estimation. In: Proc. of the 32nd Int'l Conf. on Machine Learning. Lille: JMLR.org, 2015. 881–889.

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

[207] Wu ZN, Negi P, Alizadeh M, Kraska T, Madden S. FactorJoin: A new cardinality estimation framework for join queries. Proc. of the ACM on Management of Data, 2023, 1(1): 41. [doi: 10.1145/3588721]

[208] Wang JY, Chai CL, Liu JB, Li GL. FACE: A normalizing flow based cardinality estimator. Proc. of the VLDB Endowment, 2021, 15(1): 72–84. [doi: 10.14778/3485450.3485458]

[209] Lin YM, Xu ZJ, Zhang YH, Li Y, Zhang JW. Cardinality estimation with smoothing autoregressive models. World Wide Web, 2023, 26(5): 3441–3461. [doi: 10.1007/s11280-023-01195-7]

[210] Kipf A, Kipf T, Radke B, Leis V, Boncz PA, Kemper A. Learned cardinalities: Estimating correlated joins with deep learning. arXiv:1809.00677, 2018.

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

[212] Wang F, Yan X, Yiu ML, Li S, Mao ZY, Tang B. Speeding up end-to-end query execution via learning-based progressive cardinality estimation. Proc. of the ACM on Management of Data, 2023, 1(1): 28. [doi: 10.1145/3588708]

[213] Li BB, Lu Y, Kandula S. Warper: Efficiently adapting learned cardinality estimators to data and workload drifts. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 1920–1933. [doi: 10.1145/3514221.3526179]

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

[215] Wu PZ, Cong G. A unified deep model of learning from both data and queries for cardinality estimation. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 2009–2022. [doi: 10.1145/3448016.3452830]

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

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

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

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

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

[221] Hilprecht B, Binnig C. Zero-shot cost models for out-of-the-box learned cost prediction. Proc. of the VLDB Endowment, 2022, 15(11): 2361–2374. [doi: 10.14778/3551793.3551799]

[222] Yang JN, Wu S, Zhang DX, Dai J, Li FF, Chen G. Rethinking learned cost models: Why start from scratch? Proc. of the ACM on Management of Data, 2023, 1(4): 255. [doi: 10.1145/3626769]

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

[224] Ortiz J, Balazinska M, Gehrke J, Keerthi SS. Learning state representations for query optimization with deep reinforcement learning. In: Proc. of the 2nd Workshop on Data Management for End-to-end Machine Learning. Houston: ACM, 2018. 4. [doi: 10.1145/3209889.3209890]

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

[226] Guo RB, Daudjee K. Research challenges in deep reinforcement learning-based join query optimization. In: Proc. of the 3rd Int'l Workshop on Exploiting Artificial Intelligence Techniques for Data Management. Portland: ACM, 2020. 3. [doi: 10.1145/3401071.3401657]

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

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

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

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

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

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

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

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

[235] Woltmann L, Thiessat J, Hartmann C, Habich D, Lehner W. FASTgres: Making learned query optimizer hinting effective. Proc. of the

VLDB Endowment, 2023, 16(11): 3310–3322. [doi: 10.14778/3611479.3611528]

[236] Weng LG, Zhu R, Wu D, Ding BL, Zheng BL, Zhou JR. Eraser: Eliminating performance regression on learned query optimizer. Proc. of the VLDB Endowment, 2024, 17(5): 926–938. [doi: 10.14778/3641204.3641205]

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

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

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

[240] Sioulas P, Ailamaki A. Scalable multi-query execution using reinforcement learning. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 1651–1663. [doi: 10.1145/3448016.3452799]

[241] Wei ZY, Trummer I. SkinnerMT: Parallelizing for efficiency and robustness in adaptive query processing on multicore platforms. Proc. of the VLDB Endowment, 2022, 16(4): 905–917. [doi: 10.14778/3574245.3574272]

[242] Mao HZ, Schwarzkopf M, Venkatakrishnan SB, Meng ZL, Alizadeh M. Learning scheduling algorithms for data processing clusters. In: Proc. of the 2019 ACM Special Interest Group on Data Communication. Beijing: ACM, 2019. 270–288. [doi: 10.1145/3341302.3342080]

[243] Sabek I, Ukyab TS, Kraska T. LSched: A workload-aware learned query scheduler for analytical database systems. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 1228–1242. [doi: 10.1145/3514221.3526158]

[244] Zhang C, Marcus R, Kleiman A, Papaemmanouil O. Buffer pool aware query scheduling via deep reinforcement learning. arXiv:2007.10568, 2022.

[245] Zhang W, Chen C, Wang QG, Wang W, Yang SJ, Zhou BY, Zhu HM, Chen C, Zhao YJ, Hu YQ, Cheng MM, Li M, Tan HF, Liu MJ, Lin HX, Zhang S, Zhang L. BG3: A cost effective and I/O efficient graph database in Bytedance. In: Proc. of Companion of the 2024 Int'l Conf. on Management of Data. Santiago: ACM, 2024. 360–372. [doi: 10.1145/3626246.3653373]

[246] Zeng TJ, Wei ZW, Luo G, Yi K, Du XY, Wen JR. Persistent summaries. ACM Trans. on Database Systems (TODS), 2022, 47(3): 11. [doi: 10.1145/3531053]

[247] Sun ZY, Zhou XH, Li GL. Learned index: A comprehensive experimental evaluation. Proc. of the VLDB Endowment, 2023, 16(8): 1992–2004. [doi: 10.14778/3594512.3594528]

[248] Bachfischer M, Borovica-Gajic R, Rubinstein BIP. Testing the robustness of learned index structures. arXiv:2207.11575, 2022.

[249] Ferragina P, Lillo F, Vinciguerra G. Why are learned indexes so effective? In: Proc. of the 37th Int'l Conf. on Machine Learning. JMLR.org, 2020. 293.

[250] Al-Mamun A, Wu H, He QY, Wang JG, Aref WG. A survey of learned indexes for the multi-dimensional space. arXiv:2403.06456, 2024.

[251] Liu QY, Li MC, Zeng YX, Shen YY, Chen L. How good are multi-dimensional learned indexes? An experimental survey. The VLDB Journal, 2025, 34(2): 17. [doi: 10.1007/s00778-024-00893-6]

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

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

[254] Sun J, Zhang JT, Sun ZY, Li GL, Tang N. Learned cardinality estimation: A design space exploration and a comparative evaluation. Proc. of the VLDB Endowment, 2021, 15(1): 85–97. [doi: 10.14778/3485450.3485459]

[255] Heinrich R, Luthra M, Wehrstein J, Kornmayer H, Binnig C. How good are learned cost models, really? Insights from query optimization tasks. Proc. of the ACM on Management of Data, 2025, 3(3): 172. [doi: 10.1145/3725309]

[256] Lehmann C, Sulimov P, Stockinger K. Is your learned query optimizer behaving as you expect? A machine learning perspective. Proc. of the VLDB Endowment, 2024, 17(7): 1565–1577. [doi: 10.14778/3654621.3654625]

[257] Lim WS, Butrovich M, Zhang W, Crotty A, Ma L, Xu PJ, Gehrke J, Pavlo A. Database Gyms. 2023. https://www.cidrdb.org/cidr2023/slides/p27-lim-slides.pdf

[258] Zhu R, Weng LG, Wei WQ, Wu D, Peng JZ, Wang YF, Ding BL, Lian DF, Zheng BL, Zhou JR. PilotScope: Steering databases with machine learning drivers. Proc. of the VLDB Endowment, 2024, 17(5): 980–993. [doi: 10.14778/3641204.3641209]

附中文参考文献

[1] 孙路明, 张少敏, 姬涛, 李翠平, 陈红. 人工智能赋能的数据管理技术研究. 软件学报, 2020, 31(3): 600–619. http://www.jos.org.cn/1000-9825/5909.htm [doi: 10.13328/j.cnki.jos.005909]

[131] 金连源, 李国良. 基于人工智能方法的数据库智能诊断. 软件学报, 2021, 32(3): 845–858. http://www.jos.org.cn/1000-9825/6177.htm [doi: 10.13328/j.cnki.jos.006177]

[155] 李奕言, 田季坤, 蒲照, 李翠平, 陈红. 数据库参数配置智能调优研究综述. 计算机学报, 2024, 47(8): 1901–1921. [doi: 10.11897/SP.J.1016.2024.01901]

作者简介

姬涛, 博士生, 主要研究领域为人工智能与数据库, 学习型索引, 索引推荐, 查询优化.

钟锴, 博士生, 主要研究领域为人工智能与数据库, 查询优化.

李奕言, 博士生, 主要研究领域为人工智能与数据库, 数据库参数调优.

李翠平, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为机器学习与数据库, 查询优化.

陈红, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为机器学习与数据库, 数据库与高性能计算, 隐私保护.