**2023“CCF-蚂蚁科研基金 数据库实验室专项”**

**数据库索引智能推荐和管理技术研究**

**申报课题：分布式数据库系统中自适应的索引推荐算法**

|  |  |
| --- | --- |
| **申请者姓名** | 周旭 |
| **CCF会员码** | 99840S |
| **所属机构** | 湖南大学 |
| **提交日期** | 2023-8-22 |

**目录**

**1. 项目介绍**

1.1 基金题目

1.2 背景及研究意义

1.3 研究目标

1.4 研究方法

**2. 研究计划**

2.1 项目期限

2.2 项目计划

**3. 预期成果**

**4. 人力、设备等投入及项目预算**

4.1 人力投入

4.2 设备投入

4.3 项目总预算

**5. 教授个人简历**

# 项目介绍

## 1.1 基金题目

**数据库索引智能推荐和管理技术研究**

## 1.2 背景及研究意义

### 1.2.1 研究背景与意义

近年来，随着互联网应用的不断发展与日渐成熟，在云计算、大数据、人工智能等新技术的带动下，数据量以惊人的速度不断膨胀，通过提供用户友好的声明式查询范例以及封装复杂的查询优化函数，数据库技术在各个行业中发挥着十分重要的作用。索引是目前对海量数据进行高效查询的主要方法，它记录了数据属性（被称为索引的搜索键）到数据记录间的映射关系，在查询执行过程中，通过查询索引表的方式来代替全表扫描，可以大幅缩短查询时间。因此，如何在分布式数据库系统中进行索引的自适应推荐和管理是当前数据库研究领域的一个热点问题。

作为全球唯一在TPC-C和TPC-H测试上都刷新了世界纪录的国产原生分布式数据库， OceanBase由蚂蚁集团完全自主研发，采用自研的一体化架构，兼顾分布式架构的扩展性与集中式架构的性能优势，OceanBase使用一套引擎同时支持OLTP和OLAP的混合负载，具有数据强一致、高可用、高性能、在线扩展、高度兼容SQL标准和主流关系数据库、对应用透明、高性价比等特点。

OceanBase在同一个分布式数据库上能同时支持OLTP和OLAP两种负载场景，避免了在传统架构中在线与离线数据库之间大量的数据传输。在目前分布式数据库使用中，一些经验欠佳的用户可能会建立大量的索引，然而其中大部分的索引在实际中并没有被使用，但这些索引占用了存储和计算资源。因此研究如何针对候选索引进行选择和推荐是非常有必要的。

传统的索引推荐方法大多采用启发式规则进行枚举和贪心。在负载动态变化的生产环境数据库中，枚举算法无法在有限的时间内计算出所有索引组合情况。基于贪心算法的索引推荐方法，可能存在计算出次优解的问题。很多数据库供应商提供索引推荐工具来帮助数据库管理人员减轻工作负担，但是这需要数据库管理员有丰富的使用经验。而机器学习可以通过学习历史数据中的关系获得经验，辅助数据库系统做决策，这为数据库系统中的索引推荐技术提供了新的可能性，同时也给索引推荐带来了一些新的挑战：

1. **如何估计索引的收益？**通用的方式是将索引信息添加到数据库管理系统的数据字典中，而不是创建实际的索引，以便数据库管理系统中的查询优化器可以识别索引的存在，并在不构建实际索引的情况下估计查询执行的成本。在这种情况下，索引的收益估算其实是查询执行的估计成本在有假想索引和无假想索引的情况下的减少量。
2. **如何对索引进行选择?** 基于强化学习的索引选择方法是随着强化学习的技术发展起来的，与传统索引选择方法相比，它可以在数据库负载动态变化的环境下进行索引选择，不需要借助数据库管理员的经验。基于强化学习的索引选择过程本质是马尔可夫过程，通过神经网络学习使用不同索引前后对负载的代价影响，之后根据已经创建的索引预测下一个应该建立索引的概率，最终根据不同的负载选择出合适的索引。
3. **如何高效地更新索引以保证其高性能?** 面对这个挑战，需要面向动态工作负载的增量索引管理系统，通过输入即将到来的查询和历史索引统计信息，在资源受限的情况下更新现有索引集，以提升性能(如更高的吞吐量)，并基于已有索引和候选索引利用蒙特卡罗树搜索选择高效益的索引，同时可以训练一个深度回归模型来估计整体的索引效益。

针对以上问题，本项目拟从索引收益评估、索引的选择和索引集合的更新三个方面，结合人工智能有关技术，研究分布式环境下的索引推荐及管理系统，通过提出相应的索引推荐策略，来减少候选索引的数量，同时减少不必要的空间开销，提升查询性能，从而提升数据库索引的质量。

### 1.2.2 国内外相关工作

数据库中的自动索引推荐一直是近期的热点研究领域[1, 2]，也是提高数据库查询性能的关键。索引的收益估计也就是对对索引结构[3-7]和成本模型[8,9]的研究。该问题最初被定义为在指定工作负载上搜索索引的问题[10,11]，许多研究利用被称为“what-if” API（使用自定方案假设索引的配置）[12]的查询优化器的扩展来解决这个问题。随后，部分研究使用不同的搜索策略[13] 使该问题形成了几个变体，将索引推荐与其他物理设计结构（如分区、物化视图或列存储）集成在一起[14-17]，将其制定为连续调优问题[18, 19]，或者物理设计的鲁棒性建模[20]。所有这些方法都专注于使用查询优化器的成本估计进行推荐，但是许多回归模型的错误是由查询优化器引起的，改进优化器也一直是一个活跃的研究领域[21-25]，正如在[26,27]论文中所观察到的，查询优化中的估计误差仍然是一个显著问题。最近，Pavlo等[28]和Sharma等[29]使用先进的ML技术，如神经网络或强化学习来自动化索引调优，但由于只提供了视觉而缺少细节。本项目中提出的索引收益估计，关注于从执行成本方面提高推荐质量，以及利用现有技术快速适应未见数据。

在索引选择问题(Index Selection Problem, ISP)方面，人们提出了几种传统的方法来为工作负载寻找合适的索引[30-34]。Chaudhuri 等 [30]和Valentin 等[34]提出了一种基于贪婪的两阶段方法。但是，文献[30]中没有考虑不同索引之间的交互作用，如果索引a的收益受到索引b的影响，那么索引a和索引b之间就存在交互作用，文献[35]则反之。虽然在[34]中考虑了这种交互，但它只是在搜索可能更好的索引时随机地进行多次索引的打乱。ILP等[32]和Cophy等[31]将ISP建模为一个二进制整数问题(BIP)，并采用了一个商业的BIP求解器。然而，这些方法没有考虑不同索引之间的交互，假设查询中的每个表只能使用一个索引。通过多个索引访问表是一种特殊的索引交互形式，大多数商业和开源数据库都支持这种访问方法。Schlosser等人在文献[33]中提出了一种不生成候选索引的单阶段方法。在每个步骤中，选择一个新的单一属性索引，或者通过添加新属性来扩展现有的索引，以最大化每个额外内存的额外性能。然而，它们从选择单属性索引开始，从而错过了多属性索引可以带来的更大的性能提升，此外，它们只考虑基于已建索引的交互。在此之后，一些基于学习的方法也被提出来解决ISP [36-39]。Sadri等人[37]在ISP中为集群数据库使用深度强化学习(DRL)。然而，他们只推荐单属性索引，并且没有实现或评估他们的方法。文献[39]中的方法没有考虑约束，仅处理涉及一张表的简单查询。文献[38]中的方法仅处理涉及一张表的查询，并建立单属性索引。Ding等[36]使用神经网络来比较不同索引配置下的负载开销，而不是使用what-if调用者。然而，该算法的其他部分仍然采用了文献[30,34]中的方法。因此，它仍然未能很好地对交互进行建模。**本项目将ISP描述为DRL问题，并提出了一种两阶段的方法。直观地说，与传统的基于贪婪的方法相比，DRL可以探索更多的组合，这些组合可以更好地预见引入新索引的全局影响。**

在索引管理方面，现有的索引管理方法大致可以分为两类，包括基于优化器的方法和基于ML的方法。对于基于优化器的方法，它们依赖传统的数据库估计器来估计不同索引下的查询代价，并利用一些启发式算法(如top-K, hill-climbing)来选择能够降低执行代价的索引[40-45]。然而，估计的成本可能是不准确的，并会误导得到次优的解决方案。对于基于机器学习的方法，为解决成本估算不准确的问题，Ding等人[36]提出设计一种神经网络，根据指标创建前后的计划来估算指标效益，并据此推荐任何具有正效益的指标。然而，它没有考虑到(i)索引和查询之间的相关性(例如，维护成本可能比减少的查找成本更高)和(ii)删除冗余/负索引，这些会显著影响性能。在数据库优化[46-50](如加入枚举的深度强化学习[44，51-55]、视图管理[56,57]、数据分区[58])中采用机器学习的工作越来越多。然而，由于两个原因，先进的DRL方法(如DQN [56,59]，DDPG[60,61])不能直接应用于项目的问题。首先，RL基于静态的环境-角色交互，需要很长时间(如几天或几周)来适应动态的工作负载[62,63]，这在现实中是难以忍受的；其次，像DDPG这样的高级RL方法只支持在每次迭代中创建几个索引，但很难支持索引删除等其他操作，因为它不能返回到之前的步骤。

**本项目将研究一种轻量级的探索方法(lightweight exploration method)，不仅可以平衡探索和利用(寻找有希望的索引解决方案)之间的关系，而且可以维护一个策略树(代表所有索引解决方案)，帮助动态负载更新历史索引(父节点)和添加新索引(子节点)。**

## 参考文献

1. Sheldon J. Finkelstein, Mario Schkolnick, and Paolo Tiberio. 1988. Physical Database Design for Relational Databases. ACM Trans. Database Syst. 13, 1(1988), 91-128.
2. Michael Stonebraker. 1974. The choice of partial inversions and combined indices. International Journal of Parallel Programming 3, 2 (1974), 167-188.
3. Ioannis Alagiannis, Stratos Idreos, and Anastasia Ailamaki. 2014. H2O: a hands-free adaptive store. In SIGMOD. 1103–1114.
4. Joy Arulraj, Andrew Pavlo, and Prashanth Menon. 2016. Bridging the Archipelago between Row-Stores and Column-Stores for Hybrid Workloads. In SIGMOD. 583–598.
5. Stratos Idreos, Martin L. Kersten, and Stefan Manegold. 2007. Database Cracking. In CIDR. 68–78.
6. Stratos Idreos, Stefan Manegold, and Goetz Graefe. 2012. Adaptive indexing in modern database kernels. In EDBT. 566–569.
7. Stratos Idreos, Stefan Manegold, Harumi A. Kuno, and Goetz Graefe. 2011. Merging What’s Cracked, Cracking What’s Merged: Adaptive Indexing in Main-Memory Column-Stores. PVLDB 4, 9 (2011), 585–597.
8. Michael Stillger, Guy M. Lohman, Volker Markl, and Mokhtar Kandil. 2001. LEO - DB2’s LEarning Optimizer. In VLDB. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 19–28.
9. Wentao Wu, Yun Chi, Shenghuo Zhu, Jun’ichi Tatemura, Hakan Hacigümüs, and Jeffrey F. Naughton. 2013. Predicting Query Execution Time: Are Optimizer Cost Models Really Unusable?. In ICDE. IEEE Computer Society, Washington, DC, USA, 1081–1092.
10. Surajit Chaudhuri and Vivek R. Narasayya. 1997. An Efficient Cost- Driven Index Selection Tool for Microsoft SQL Server. In VLDB. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 146–15
11. Gary Valentin, Michael Zuliani, Daniel C. Zilio, Guy M. Lohman, and Alan Skelley. 2000. DB2 Advisor: An Optimizer Smart Enough to Recommend Its Own Indexes. In ICDE. 101–110.
12. Surajit Chaudhuri and Vivek R. Narasayya. 1998. AutoAdmin ‘What-if’ Index Analysis Utility. In SIGMOD. 367–378.
13. Nicolas Bruno and Surajit Chaudhuri. 2007. Physical design refinement: The ‘merge-reduce’ approach. ACM Trans. Database Syst. 32, 4 (2007), 28.
14. Sanjay Agrawal, Surajit Chaudhuri, Lubor Kollár, Arunprasad P. Marathe, Vivek R. Narasayya, and Manoj Syamala. 2004. Database Tuning Advisor for Microsoft SQL Server 2005. In VLDB. 1110–1121.
15. Adam Dziedzic, Jingjing Wang, Sudipto Das, Bolin Ding, Vivek R. Narasayya, and Manoj Syamala. 2018. Columnstore and B+ tree - Are Hybrid Physical Designs Important?. In SIGMOD. 177–190.
16. Michael S. Kester, Manos Athanassoulis, and Stratos Idreos. 2017. Access Path Selection in Main-Memory Optimized Data Systems: Should I Scan or Should I Probe?. In SIGMOD. 715–730.
17. Daniel C. Zilio, Jun Rao, Sam Lightstone, Guy Lohman, Adam Storm, Christian Garcia-Arellano, and Scott Fadden. 2004. DB2 Design Advisor: Integrated Automatic Physical Database Design. In VLDB. VLDB Endowment, 1087–1097.
18. Nicolas Bruno and Surajit Chaudhuri. 2007. An Online Approach to Physical Design Tuning. In ICDE. 826–835.
19. Karl Schnaitter, Serge Abiteboul, Tova Milo, and Neoklis Polyzotis. 2006. COLT: continuous on-line tuning. In SIGMOD. 793–795.
20. Kareem El Gebaly and Ashraf Aboulnaga. 2008. Robustness in automatic physical database design. In EDBT. 145–156.
21. Surajit Chaudhuri. 1998. An Overview of Query Optimization in Relational Systems. In PODS. ACM, New York, NY, USA, 34–43.
22. Surajit Chaudhuri. 2009. Query optimizers: time to rethink the contract?. In SIGMOD. 961–968.
23. Surajit Chaudhuri, Vivek Narasayya, and Ravi Ramamurthy. 2008. A pay-as-you-go framework for query execution feedback. PVLDB 1, 1 (2008), 1141–1152.
24. Chungmin Melvin Chen and Nick Roussopoulos. 1994. Adaptive Selectivity Estimation Using Query Feedback. In SIGMOD. ACM, New York, NY, USA, 161–172.
25. Wentao Wu, Jeffrey F. Naughton, and Harneet Singh. 2016. Sampling- Based Query Re-Optimization. In SIGMOD. ACM, New York, NY, USA, 1721–1736.
26. Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? PVLDB 9, 3 (Nov. 2015), 204–215.
27. Guy Lohman. 2014. Is Query Optimization a “Solved” Problem?
28. Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon, Todd C. Mowry, Matthew Perron, Ian Quah, Siddharth Santurkar, Anthony Tomasic, Skye Toor, Dana Van Aken, Ziqi Wang, Yingjun Wu, Ran Xian, and Tieying Zhang. 2017. Self- Driving Database Management Systems. In CIDR.
29. Ankur Sharma, Felix Martin Schuhknecht, and Jens Dittrich. 2018. The Case for Automatic Database Administration using Deep Reinforcement Learning. CoRR abs/1801.05643 (2018).
30. Surajit Chaudhuri and Vivek R. Narasayya. 1997. An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. In VLDB 1997. 146–155.
31. Debabrata Dash, Neoklis Polyzotis, and Anastasia Ailamaki. 2011. CoPhy: A Scalable, Portable, and Interactive Index Advisor for Large Workloads. Proc. VLDB Endow. 4, 6 (2011), 362–372.
32. Stratos Papadomanolakis and Anastassia Ailamaki. 2007. An Integer Linear Programming Approach to Database Design. In ICDE 2007. 442–449.
33. Rainer Schlosser, Jan Kossmann, and Martin Boissier. 2019. Efficient Scalable Multi-attribute Index Selection Using Recursive Strategies. In ICDE 2019.
34. Gary Valentin, Michael Zuliani, Daniel C. Zilio, Guy M. Lohman, and Alan Skelley. 2000. DB2 Advisor: An Optimizer Smart Enough to Recommend Its Own Indexes. In ICDE 2000. 101–110.
35. Karl Schnaitter, Neoklis Polyzotis, and Lise Getoor. 2009. Index Interactions in Physical Design Tuning: Modeling, Analysis, and Applications. Proc. VLDB Endow. 2, 1 (2009), 1234–1245.
36. Bailu Ding, Sudipto Das, Ryan Marcus, Wentao Wu, Surajit Chaudhuri, and Vivek R. Narasayya. 2019. AI Meets AI: Leveraging Query Executions to Improve Index Recommendations. In SIGMOD 2019. 1241–1258.
37. Zahra Sadri, Le Gruenwald, and Eleazar Leal. 2020. Online Index Selection Using Deep Reinforcement Learning for a Cluster Database. In ICDEW. 158–161.
38. Ankur Sharma, Felix Martin Schuhknecht, and Jens Dittrich. 2018. The Case for Automatic Database Administration using Deep Reinforcement Learning. CoRR abs/1801.05643 (2018).
39. Jeremy Welborn, Michael Schaarschmidt, and Eiko Yoneki. 2019. Learning Index Selection with Structured Action Spaces. CoRR abs/1909.07440 (2019).
40. S. Chaudhuri and V. R. Narasayya. An efficient cost-driven index selection tool for microsoft SQL server. In VLDB, pages 146–155. Morgan Kaufmann, 1997.
41. G. Valentin, M. Zuliani, D. C. Zilio, G. M. Lohman, and A. Skelley. DB2 advisor: An optimizer smart enough to recommend its own indexes. In D. B. Lomet and G. Weikum, editors, ICDE, pages 101–110. IEEE Computer Society, 2000.
42. S. Agrawal, S. Chaudhuri, L. Koll´ar, A. P. Marathe, V. R. Narasayya, and M. Syamala. Database tuning advisor for microsoft SQL server 2005. In VLDB, pages 1110–1121, 2004.
43. S. Agrawal, S. Chaudhuri, L. Koll´ar, A. P. Marathe, V. R. Narasayya, and M. Syamala. Database tuning advisor for microsoft SQL server 2005: demo. In F. O¨ zcan, editor, SIGMOD, pages 930–932. ACM, 2005.
44. M. Stillger, G. M. Lohman, V. Markl, and M. Kandil. LEO - db2’s learning optimizer. In VLDB, pages 19–28, 2001.
45. G. Li, X. Zhou, and S. Li. Xuanyuan: An ai-native database. IEEE Data Eng. Bull., 42(2):70–81, 2019.
46. X. Zhou, C. Chai, G. Li, and J. Sun. Database meets artificial intelligence: A survey. TKDE, 2020.
47. G. Li, X. Zhou, and L. Cao. AI meets database: AI4DB and DB4AI. In SIGMOD, pages 2859–2866, 2021.
48. W. Wang, M. Zhang, G. Chen, H. V. Jagadish, and et al. Database meets deep learning: Challenges and opportunities. SIGMOD Rec., 2016.
49. G. Li and X. Zhou. Machine learning for data management: A system view. In ICDE, 2022.
50. C. Lin, J. Zhuang, J. Feng, H. Li, X. Zhou, and G. Li. Adaptive code learning for spark configuration tuning. In ICDE, 2022.
51. S. Krishnan, Z. Yang, K. Goldberg, J. M. Hellerstein, and I. Stoica. Learning to optimize join queries with deep reinforcement learning. CoRR, abs/1808.03196, 2018.
52. R. Marcus and O. Papaemmanouil. Deep reinforcement learning for join order enumeration. In SIGMOD 2018, pages 3:1–3:4, 2018.
53. X. Yu, G. Li, C. chai, and N. Tang. Reinforcement learning with tree-lstm for join order selection. In ICDE 2020, pages 196–207, 2019.
54. H. Lan, Z. Bao, and Y. Peng. A survey on advancing the dbms query optimizer: Cardinality estimation, cost model, and plan enumeration. Data Science and Engineering, 6(1):86–101, 2021.
55. L. Cui, J. Chen, W. He, H. Li, W. Guo, and Z. Su. Achieving approximate global optimization of truth inference for crowdsourcing microtasks. Data Science and Engineering, 6(3):294–309, 2021.
56. H. Yuan, G. Li, L. Feng, and et al. Automatic view generation with deep learning and reinforcement learning. In ICDE, 2020.
57. Y. Han, G. Li, H. Yuan, and J. Sun. An autonomous materialized view management system with deep reinforcement learning. In ICDE, 2021.
58. B. Hilprecht, C. Binnig, and U. R¨ohm. Learning a partitioning advisor for cloud databases. In SIGMOD, 2020.
59. S. Huang, Y. Wang, T. Zhao, and G. Li. A learning-based method for computing shortest path distances on road networks. In ICDE, pages 360–371, 2021.
60. L. Zhang, C. Chai, X. Zhou, and G. Li. Learnedsqlgen: Constraint-aware sql generation using reinforcement learning. In SIGMOD, 2022.
61. G. Li, X. Zhou, S. Li, and B. Gao. Qtune: A query-aware database tuning system with deep reinforcement learning. PVLDB, 2019.
62. I. Trummer, J. Wang, D. Maram, S. Moseley, S. Jo, and J. Antonakakis. Skinnerdb: Regret-bounded query evaluation via reinforcement learning. In SIGMOD 2019, pages 1153–1170, 2019.
63. G. Li, X. Zhou, and L. Cao. Machine learning for databases. Proc. VLDB Endow., 14 (12):3190–3193, 2021.

## 1.3 研究目标

1. **研究基于机器学习的索引收益估算技术**

本项目将基于机器学习的方法，在不构建实际索引的情况下以查询时减少的成本来估算索引产生的收益。同时，本项目还将结合OceanBase的OLTP和OLAP工作负载以及历史索引的使用信息，设计一种深度回归模型，提升索引选择的准确性。

1. **提出基于强化学习的索引选择方法**

本项目将基于目标1中设计的基于机器学习的索引收益评估技术，提出结合OceanBase不同索引类型的深度强化学习索引选择算法。该算法加入负载索引扫描行数特征矩阵，并对奖励值进行归一化处理，旨在提高索引选择的准确性和效率，减少索引空间占用，在保证相当的索引推荐质量的同时，选择出占用更小存储空间的索引。

1. **研发基于机器学习的动态增量索引管理系统**

本项目拟提出一种针对动态工作负载和OceanBase高拓展性下动态分布式节点变化的增量式索引管理系统，在适应不断变化的工作负载或分布式节点数目的变化的同时提高数据库的查询性能。该系统利用候选索引生成方法来高效地捕获即将到来的工作负载的索引需求，并基于利用深度回归模型进行的索引收益估算以及蒙特卡洛树搜索算法来进行索引的选择，以在可能的索引组合中找到最优解。同时，该系统还通过增量更新索引来保证持续的高性能。

## 1.4 研究方法

本项目首先针对索引推荐系统中的收益估算问题提出了基于机器的索引收益评估技术，利用有/无索引时查询执行的成本减少量估算索引收益；然后基于强化学习提出了一种索引选择方法，同时减少了候选索引的空间占用；最后基于前两项技术，设计了基于动态工作负载和分布式节点变化的增量索引管理系统，从而实现分布式数据库系统中自适应的索引推荐与管理，具体研究方法如下：

1. 基于**机器学习的索引收益评估技术**

对于数据库系统来说，自动索引实施的一个关键要求是，创建或删除索引不会导致明显的查询单位性能回归。这种回归（即改变索引后查询的执行成本增加）是全自动索引的一个主要障碍，因为用户希望强制执行无查询回归约束。目前最先进的工业级索引调优系统依赖于查询优化器的成本估算，推荐具有最大估计改进的索引。当调优器搜索备选索引配置时，需要比较同一查询对应不同配置的不同计划的执行成本。调优器可以使用优化器的估计值来强制执行无回归约束。然而，由于优化器的估计值存在局限性，例如在卡方估计或成本模型中存在误差，因此使用优化器的估计值来强制执行约束可能会导致重大误差。

由于上述的调优器可能带来误差的原因，项目拟利用机器学习技术来提高索引推荐的质量。比较两个计划的执行成本可以被表述为机器学习中的分类任务。训练分类器来决定一对计划中哪个计划的执行成本更低，与使用（学习或分析）成本模型来比较成本相比，准确率更高。如果准确的话，与分类方法相比，这种模型的通用性要高得多。

OceanBase数据库拥有一套独特的SQL执行引擎，可以同时支持混合OLTP和OLAP的工作负载，因此当评估索引收益的时候，项目拟使用以下指导原则来编码查询计划的语义和在特征化中导致执行成本的其他因素：

1. **从OceanBase数据库的OLAP/OLTP工作负载中学习**
2. **从历史索引使用信息中学习**
3. **从优化器中学习**
4. **从估计查询计划中的信息中学习。**

除上述原则外，特征向量还应编码以下几类关键信息，为模型学习分类任务提供途径：

1. **已完成工作的度量。**查询优化器对操作符成本的估计或操作符处理的行数是这种度量的示例特征。
2. **结构信息。**连接命令或操作符在计划中的位置通常很有用，尤其是在比较同一查询的两个计划时。
3. **物理操作符详情。**计划中的物理操作符对成本起着至关重要的作用。例如，嵌套循环连接与合并连接相比，即使它们对应于相同的逻辑连接操作符，成本也会大不相同。

对于给定的查询计划，项目为每个键赋值：(i) 衡量计划中相应操作符所做的工作量；(ii) 编码结构信息。在多个操作符具有相同键的计划中，项目会将分配给该键的所有值相加。如果一个运算符在计划中没有出现，项目就把对应的键值赋值为零，这样就可以得到一个固定维度的向量。为操作符赋值的不同方式编码了不同的信息，并创建了不同的特征通道。

|  |  |
| --- | --- |
| **通道** | **描述** |
| *EstNodeCost* | 节点估计成本作为节点权重 |
| *EstRowProcessed* | 节点处理的估计行数作为其权重 |
| *EstBytesProcessed* | 节点处理的估计字节数作为其权重 |
| *EstRows* | 节点输出的估计行数作为其权重 |
| *EstBytes* | 节点输出的估计字节数作为其权重 |
| *LeafWeightEst-RowsWeightedSum* | 估计行数作为叶片权重，权重总和作为节点权重 |
| *LeafWeightEst-BytesWeightedSum* | 估计字节数作为叶片权重，权重总和作为节点权重 |

表1 特征通道的样例

表中列出了不同的特征通道、权重计算方式以及它们所编码的信息。每个通道的维度相同。由于表中的通道有一定的冗余度，因此只要项目选择的通道能编码一定程度的工作和结构信息，那么一个通道子集（通常是两个或三个）就足够了。项目还将优化器估算的计划成本作为一个特征。

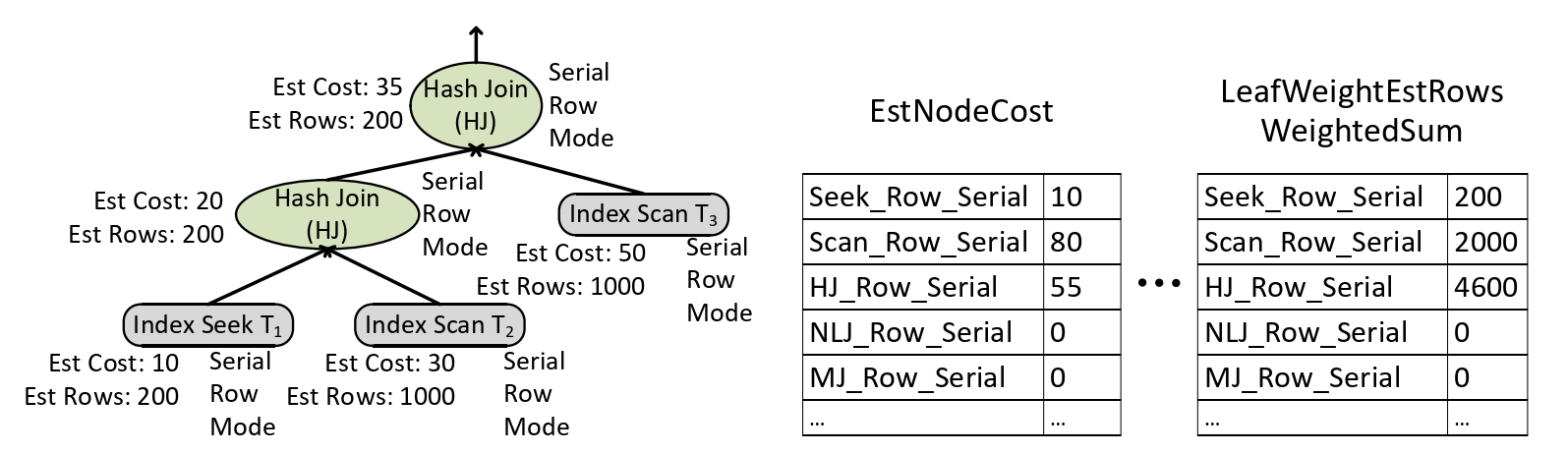


图1(a) 样例查询计划 图1（b）样例计划的特征通道

图(a)是一个样例查询计划，图(b)是图（a）所示简单查询计划进行特征化的一个示例。项目拟使用不同索引下的查询计划+成本作为训练数据，训练分类模型时，输入为带/不带索引的两个查询计划，输出为0/1，分别代表无/有提升。

在线性学习器中，项目拟选择了逻辑回归（Logistic Regression，LR），因为它简单且训练速度快。在基于树的技术中，项目准备尝试两大类模型，它们都是树的集合：(a)袋状集合，如随机森林（Random Forest，RF）；(b)提升集合，如梯度提升树（Gradient-boosted Trees，GBT）和带有梯度提升决策树的轻型 GBM（Light GBM with gradient-boosted decision trees，LGBM）。

项目将数据分成两个分离集Dl和Dm，用Dl训练本地模型（Mlocal），用Dm训练元模型（Mmeta）；离线模型Moffline是用其他数据库的执行数据训练的。项目为Dm中的每个数据点d提取元特征，如Moffline和 Mlocal对d的预测、相应的不确定性得分以及Dl中d的近邻的距离和标签。

1. **基于强化学习的索引选择方法**

OceanBase 数据库支持局部索引和全局索引。分区表的局部索引和非分区表的索引类似，索引的数据结构还是和主表的数据结构保持一对一的关系，但由于主表已经做了分区，主表的每一个分区都会有自己单独的索引数据结构。对每一个索引数据结构来说，里面的键（Key）只映射到自己分区中的主表数据，不会映射到其它分区中的主表，因此这种索引被称为局部索引。从另一个角度来看，这种模式下索引的数据结构也做了分区处理，因此有时也被称为局部分区索引（Local Partitioned Index）。

全局索引和分区表的局部索引相比，分区表的全局索引不再和主表的分区保持一对一的关系，而是将所有主表分区的数据合成一个整体来看，索引中的一个键可能会映射到多个主表分区中的数据（当索引键有重复值时）。更进一步，全局索引可以定义自己独立的数据分布模式，既可以选择非分区模式也可以选择分区模式；在分区模式中，分区的方式既可以和主表相同也可以和主表不同。因此，全局索引又分为以下两种形式：

* 全局非分区索引（Global Non-Partitioned Index）索引数据不做分区，保持单一的数据结构，和非分区表的索引类似。但由于主表已经做了分区，因此会出现索引中的某一个键映射到不同主表分区的情况，即一对多的对应关系。全局非分区索引的结构如下图所示。
* 全局分区索引（Global Partitioned Index）索引数据按照指定的方式做分区处理，例如做哈希（Hash）分区或者范围（Range）分区，将索引数据分散到不同的分区中。但索引的分区模式是完全独立的，和主表的分区没有任何关系，因此对于每个索引分区来说，里面的某一个键都可能映射到不同的主表分区（当索引键有重复值时），索引分区和主表分区之间是多对多的对应关系。全局分区索引的结构如下图所示。

除了局部索引和全局索引，OceanBase还支持行存索引，列存索引等，因此，当利用强化学习进行索引推荐的选择时，必须要考虑索引的类型和表的分区类型。

项目拟整合索引推荐规则和深度强化学习，根据不同的分区（local/global）类型或索引类型创建索引，从而得到最终的一批候选索引。

在解决了候选索引的创建后，项目将索引选择问题映射到马尔可夫决策过程中(Markov Decision Process)，其中动作st，奖励函数at，策略p(at+1|st)，智能体agent定义如下。

**State：**模型的状态表示是一个神经网络矩阵，主要包括索引的选择度矩阵 IS，索引的扫描行数特征矩阵 IR 以及索引的负载代价矩阵 IC 。状态表示索引效果好坏的信息。

**Action：**动作矩阵的长度由候选索引的个数的决定，1代表创建了此索引，0表示没有创建此索引。

**Reward：**奖励函数用来评价索引提升的性能。在第t次时间步，工作负载W，索引配置I时，第t次时间步的奖励函数定义如下：

由于不同索引对 SQL 的提升效果差距可能很大，本项目对奖励值进行了归一化，将其限定在一定的范围内，以消除奇异值的影响。最终奖励函数如下：

其中表示创建所有候选索引工作负载的代价，表示没有创建任何候选索引工作负载的代价。

**Agent：**本项目的智能体由演员网络和评论家网络组成，两个网络的结构相同，使用具有三层的完全连接的神经网络，算法结构图如图所示。

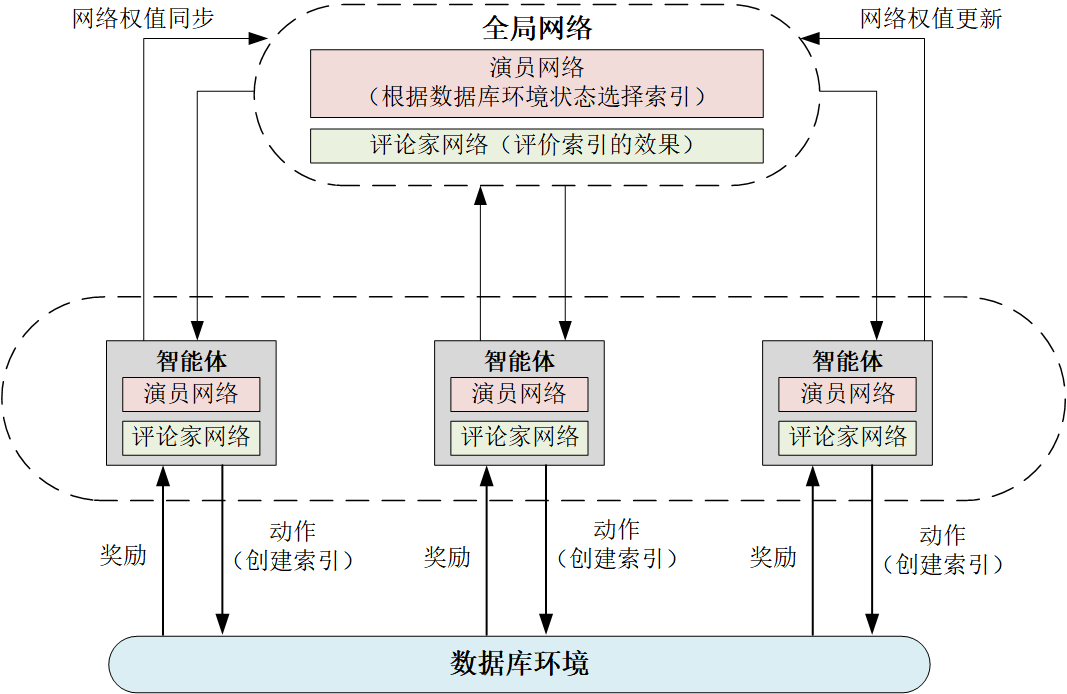


图2 优势-演员-评论家算法结构图

演员网络用来更新学习模型的策略，计算在状态所采取的动作的概率分布。演员网络可以针对给定的状态产生最佳动作，即根据状态神经网络矩阵产生最佳的索引配置。

评论家网络用来评价状态执行动作的优劣，评价索引配置的效果。其中，算法的优势函数定义如下：

其中表示即时奖励，为折扣因子，代表未来奖赏对于累计奖赏的重要程度，当时，其为1步回报优势函数，当时，其为步回报优势函数。该算法的演员网络和评论家网络的损失函数如下所示：

其中，R表示智能体在当前状态下依据策略选择动作所获得的回报值，表示该状态下的值函数，表示优势函数。

本项目首先对负载进行语法解析，并结合解析根据OceanBase数据库支持的索引类型和分区类型生成索引候选项，然后对索引候选项进行编码，转换成强化学习的动作状态，然后通过访问数据库，获取负载的特征信息、负载代价、索引扫描行数、索引存储空间，同时根据索引候选项的信息计算出索引的选择度，将这些信息作为强化学习算法的状态信息。在这些处理完成后开始进行模型的训练过程，直到模型最终收敛。该模型将利用A2C算法的并发性，减少强化学习算法的训练时间，在环境状态中加入索引扫描行数矩阵，减少推荐索引空间占用。

1. **面向动态工作负载的增量式索引管理系统**

数据管理系统中的索引可以加快数据检索速度，但代价是索引维护和存储的开销，这会严重影响工作负载性能。索引管理的目的是正确地创建或删除索引以确保索引大小在存储约束范围内以及工作负载性能最优化。在OceanBase数据库中，存在大量的工作负载查询，这些查询对索引有不同的要求（例如，访问不同的列）。并且，作为一个混合shared nothing和shared everying架构的数据库，OceanBase同时支持单机和分布式集群，兼顾分布式架构的扩展性和集中式架构的性能优势。当用户增加或删除分布式的节点时，索引也需要进行更新和管理。因此，在动态工作负载/变化分布式节点下的索引管理是很重要的。

然而，现有的管理方法主要面临三大挑战：

1. **如何有效捕获索引需求（C1）**
2. **如何有效更新索引以确保高性能（C2）**
3. **如何估算读取和写入查询的索引产生的收益（C3）**

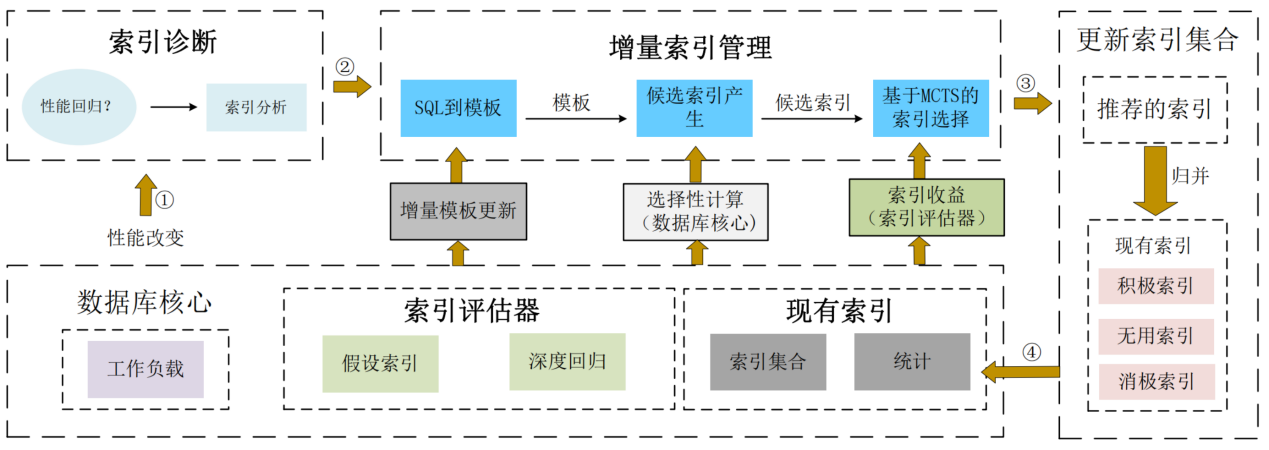


图3 系统架构

面对上述的三种挑战，项目拟设计一个基于动态工作负载的增量索引管理系统(如图)，首先，系统会监控工作负载性能，并在索引问题导致性能下降时发出索引更新请求，系统还会在OceanBase分布式节点数量发生变化的时候，发出索引更新请求。其次系统将即将到来的工作负载或者变化后的节点情况与模板相匹配，并从模板中提取有前途的候选索引（针对 C1），并利用蒙特卡洛树搜索(Monte Carlo Tree Search，MCTS)，在现有索引和候选索引（针对 C2）的基础上选择高收益索引。在蒙特卡洛树搜索中，为了有效估算不同索引的优势，系统通过训练深度回归模型来估算整体索引收益（根据读取和写入查询）（针对 C3）。

对于数据库中正在执行的任何新工作负载，当性能出现倒退时，项目首先会诊断索引问题。如果发现任何索引问题，项目会从工作负载查询（记录在运行索引管理流程的服务器中）中生成候选索引，并利用蒙特卡洛树搜索在资源限制条件下探索候选索引和现有索引的最佳组合。在蒙特卡洛树搜索中，引用项目之前提出的索引收益估算方法，该方法可根据读取和写入查询估算索引效益。最后，项目用推荐索引更新现有索引集，并根据索引效益估算结果找出冗余索引或负收益索引。

索引诊断模块在工作负载执行过程中监控系统指标。该模块一旦检测到异常状态（如性能下降），就会调用索引分析组件，以决定是否需要更新现有索引（如删除冗余索引、创建有益索引）。例如，项目计算了三类索引的比率，即：（1）未创建的有益索引；（2）很少使用的索引；（3）对工作负载性能有负面影响的索引。如果这些索引的比率高于阈值，项目将向索引推荐模块发出索引调整请求。

对于任何索引调整请求，索引推荐模块都会输入工作负载和索引统计信息，并输出推荐索引。这部分包括三个组件：SQL2Template、候选索引生成和索引选择。首先，由于工作负载可能很大（如数百万个查询），在工作负载级别推荐索引的成本很高，因此项目使用 SQL2Template 将工作负载查询映射到固定数量的查询模板中，这些查询模板表示最常用的访问模式集。对于每个查询模板，项目使用候选索引生成来解析查询子句（如 FROM、WHERE、GROUP 和 ORDER）中的谓词，并根据每个谓词涉及的列生成候选索引。例如，对于谓词 "a=$and b>$"，项目将在(a, b)上生成一个候选索引。最后，有了候选索引，项目将会利用索引选择功能，根据当前工作量推荐最佳索引。特别是，项目将会在现有索引的基础上维护一棵策略树，并在此基础上通过添加候选索引或删除现有索引等操作，探索新的最佳索引。

此外，正如索引选择方法中所述，由于OceanBase数据库支持不同的表分区类型和索引类型，项目还将支持针对数据分区情况选择索引类型，这对提高工作负载性能至关重要（例如，"全局 "索引查找速度快，但占用存储空间大；"局部 "索引效率低，但占用空间小）。

一般来说，现有数据库无法估算索引维护成本，而索引维护成本涉及复杂因素（如页面分割策略），对于估算带有不同索引的工作负载的整体性能至关重要。因此，在索引收益估算模块中，项目采用之前设计的一个深度回归模型，该模型输入工作负载特征和索引（以及索引统计信息），并输出工作负载的执行成本。该模型通过大量历史索引管理数据进行训练，可以有效地分散不同索引下的查询成本。

**2. 研究计划**

## 2.1 项目期限：以实际合同签字生效日期为准，合同期建议 1 年。

## 2.2 项目计划：

|  |  |  |  |
| --- | --- | --- | --- |
| **阶段** | **研究内容** | **时间** | **提交结果** |
| *1* | *基于机器学习的索引收益估算技术* | *第一季度(2023.10-2024.01)* | *申请发明专利1项* |
| *2* | *基于强化学习的索引选择方法* | *第二季度(2023.01-2024.04)* | *发表一篇CCF-A类论文* |
| *3* | *面向动态工作负载的增量式索引管理技术* | *第三季度(2023.04-2024.07)* | *申请发明专利1项* |
| *4* | *面向动态工作负载的增量式索引管理系统* | *第四季度(2023.071-2024.10)* | *源代码、项目报告书* |

**3. 预期成果**

1. 算法原型及源码：基于机器学习的索引收益估算算法、基于强化学习的索引选择算法及相关源代码
2. 论文：发表蚂蚁认可的CCF-A类或者领域内顶级会议、期刊论文1篇以上
3. 专利：申请国家发明专利2项以上
4. 技术指标：与现有的索引管理系统相比，支持增量索引管理，减少候选索引数量60%～70%，减少索引存储空间30%～40%。
5. **人力、设备等投入及项目预算**

## 4.1 人力投入（包括全部参与项目的人员）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **人员姓名** | **所属机构与职务** | **人员工资**  **（元/月）** | **参与项目时间**  **(折合全时人月数)** | **人员工资总计**  **（元）** | **承担职责** | **来蚂蚁实习时间（学生）** |
| *周旭* | *湖南大学 副教授* | *\*\*\*\*\*\** | *8* | *\*\*\*\*\*\** | *PI* |  |
| *张嘉鹏* | *湖南大学 助理教授* | *\*\*\*\*\*\** | *8* | *\*\*\*\*\*\** | *项目主要成员* |  |
| *石慜* | *湖南大学 博士生* | *2500* | *12* | *30000* | *项目主要成员* | *1-3* |
| *潘东* | *湖南大学 博士生* | *2500* | *12* | *30000* | *项目主要成员* | *1-3* |
| *黄阳* | *湖南大学 博士生* | *2500* | *12* | *30000* | *项目主要成员* | *1-3* |
| *张琪* | *湖南大学 研究生* | *1500* | *12* | *18000* | *项目主要成员* | *3-5* |
| *陆泽瑜* | *湖南大学 研究生* | *1500* | *12* | *18000* | *项目主要成员* | *3-5* |

## 4.2 设备投入

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **设备名称** | **数量** | **价格（元）** | **总计（元）** | **备注** |
|  |  |  |  |  |

## 4.3 项目预算

|  |  |  |
| --- | --- | --- |
| **类别** | **总计（元）** | **备注** |
| ① **人员支出** | 126000 | 与4.1对应 |
| ② **研发支出** | 144000 |  |
| 出差/会议费用 | 40000 | 1. 国内差旅单趟5000，国际差旅单趟15000，以实际开销为准； 2. 项目人员入职蚂蚁期间的津贴、补助、保险等全部费用包含在此项经费内。 |
| 设备费 |  |  |
| 其它费用 | 104000 | 由项目PI根据项目情况来填 |
| ③ **直接费用** | 270000 | ①+② |
| ④ **间接费用** | 30000 | 学校学院管理费**7**%，专票税费**3**% |
| **总计** | 300000 | ③+④ |

**5. 教授个人简历**

申请人来自湖南大学国家超级计算长沙中心、高性能计算应用软件技术教育部工程研究中心，目前所在团队共有研究人员20余人，其中国家千人计划特聘教授、 教育部长江学者和青年拔尖人才各1人、国家杰青2人。**申请人周旭副教授是超算与人工智能融合计算教育部重点实验室副主任、湖南省杰青项目获得者、湖南大学岳麓学者和博士生导师。**近年来申请人主持国家自然科学基金面上或青年基金项目3项，国家重点研发计划课题(子任务)、国家工业互联网创新发展工程和博士后特别资助项目各1项，主要从事大数据并行与分布式处理领域研究。

**在国内外权威期刊IEEE TKDE/ TPDS、《计算机学报》、《软件学报》和会议IEEE ICDE等发表/录用论文56篇，其中中国计算机学会 (CCF)推荐的A类期刊/会议24篇(第一/通信作者16篇，CCF A 会 6篇)。申请国家发明专利24项，其中已授权15项。**学术成果得到了新加坡南洋理工大学、澳大利亚新南威尔士大学和国防科技大学等国内外知名高校领域专家的广泛关注，被包括欧洲科学院院士和ACM/IEEE Fellow在内的上百名国内外著名学者正面引用和评价。

以下将重点介绍项目申请人及主要成员相关的学术论文、国家发明专利等方面的成果。

* **国内外权威期刊/会议论文：**

1. Tongfeng Weng, Xu Zhou, Yixiang Fang, Kian-Lee Tan, Kenli Li:Finding Top-k Important Edges on Bipartite Graphs: Ego-betweenness Centrality-based Approaches. ICDE 2023: 2415-2428
2. Tianyue Ren, Xu Zhou, Kenli Li, Yunjun Gao, Ji Zhang, Keqin Li:Efficient Cross Dynamic Task Assignment in Spatial Crowdsourcing. ICDE 2023: 1420-1432
3. Tongfeng Weng, Xu Zhou, Kenli Li, Kian-Lee Tan, Keqin Li:Distributed Approaches to Butterfly Analysis on Large Dynamic Bipartite Graphs. IEEE Trans. Parallel Distributed Syst. 34(2): 431-445 (2023)
4. Wensheng Luo, Xu Zhou, Kenli Li, Yunjun Gao, Keqin Li:Efficient Influential Community Search in Large Uncertain Graphs. IEEE Trans. Knowl. Data Eng. 35(4): 3779-3793 (2023)
5. Zemin Tang, Qi Xiao, Xu Zhou, Yangfan Li, Cen Chen, Kenli Li:Learning discriminative multi-relation representations for multimodal sentiment analysis. Inf. Sci. 641: 119125 (2023)
6. Yuan Xie, Yongheng Wang, Kenli Li, Xu Zhou, Zhao Liu, Keqin Li:Satisfaction-aware Task Assignment in Spatial Crowdsourcing. Inf. Sci. 622: 512-535 (2023)
7. **周旭**, Shiting Liang, Kenli Li, Yunjun Gao, Keqin Li. Bilateral Preference-aware Task Assignment in Spatial Crowdsourcing. *ICDE* 2022: 1687-1699. (CCF A)
8. **周旭**, Kenli Li, ZhiBang Yang, Keqin Li. Finding Optimal Skyline Product Combinations under Price Promotion. *IEEE Transactions on Knowledge and Data Engineering*, 2019, 31(1): 138-151. (CCF A)
9. **周旭**, Kenli Li, ZhiBang Yang, Guoqing Xiao, Keqin Li. Progressive Approaches for Pareto Optimal Groups Computation. *IEEE Transactions on Knowledge and Data Engineering*, 2019, 31(3): 521-534. (CCF A)
10. **周旭**, Kenli Li, Yantao Zhou, Keqin Li. Adaptive Processing for Distributed Skyline Queries over Uncertain Data. *IEEE Transactions on Knowledge and Data Engineering*, 2016, 28(2): 371-384.
11. **周旭**, Kenli Li, Guoqing Xiao , Yantao Zhou, Keqin Li. Top k Favorite Probabilistic Products Queries. *IEEE Transactions on Knowledge and Data Engineering*, 2016, 28(10): 2808-2821.
12. Wensheng Luo, Kenli Li, **周旭（通信作者）**, Yunjun Gao, Keqin Li. Maximum Biplex Search over Bipartite Graphs. *ICDE* 2022: 898-910. (CCF A)
13. Yuanyuan Zeng, Wangdong Yang**（通信作者）**, **周旭（通信作者）**, Guoqing Xiao, Yunjun Gao, Kenli Li. Distributed Set Label-Constrained Reachability Queries over Billion-Scale Graphs. *ICDE* 2022: 1969-1981. (CCF A)
14. 黄阳, **周旭（通信作者）**, 杨志邦, 余婷, 张吉, 曾源远, 李肯立. 基于缓存的时变道路网最短路径查询算法. *计算机研究与发展*, 2022, 59(2): 376-389. (CCF 中文A)
15. Yuanyuan Zeng, Kenli Li, **周旭（通信作者）**, Wensheng Luo, Yunjun Gao. An Efficient Index-Based Approach to Distributed Set Reachability on Small-World Graphs. *IEEE Transactions on Parallel and Distributed Systems*, 2021, 33(10): 2358-2371. (CCF A)
16. Tongfeng Weng, **周旭（通信作者）**, Kenli Li, Peng Peng, Keqin Li. Efficient Distributed Approaches to Core Maintenance on Large Dynamic Graphs. *IEEE Transactions on Parallel and Distributed Systems*, 2021, 33(1): 129-143. (CCF A)
17. **周旭**, 翁同峰, 杨志邦, 李博仁, 张吉, 李肯立. 面向大规模二部图的分布式Tip分解算法. *软件学报*, 2021, 33(3): 0-0. (CCF 中文A)
18. Wensheng Luo, **周旭（通信作者）**, Kenli Li, Yunjun Gao, Keqin Li. Efficient influential community search in large uncertain graphs. *IEEE Transactions on Knowledge and Data Engineering*, 2021. (CCF A)
19. Wensheng Luo, **周旭（通信作者）**, Jianye Yang, Peng Peng, Guoqing Xiao, Yunjun Gao. Efficient Approaches to Top-r Influential Community Search. *IEEE Internet of Things Journal*, 2020, 8(16): 12650-12657. (SCI 1 区, IF=9.515, 稿件编号: IoT-10046-2020, 已录用, 一作为指导的博士生)
20. Kai Zhong, **周旭（通信作者）**, Liqian Zhou, Zhibang Yang, Chubo Liu, Na Xiao. Efficient Parallel Direction-based Clustering Algorithm for Large Data Sets. *Journal of Parallel and Distributed Computing*, 2020, 145: 24-33. (一作为指导的博士生)
21. **周旭**, Kenli Li, ZhiBang Yang, Yunjun Gao, Keqin Li. Efficient Approaches to k Representative G-Skyline Queries. *ACM Transactions on Knowledge Discovery from Data*, 2020, 14(5): 58:1-58:27.
22. [Zhao Liu](https://dblp.uni-trier.de/pid/99/6737.html), Kenli Li, [**Xu Zhou**](https://dblp.uni-trier.de/pid/66/5686.html), [Ningbo Zhu](https://dblp.uni-trier.de/pid/123/3661.html), [Keqin Li](https://dblp.uni-trier.de/pid/l/KeqinLi.html). Incentive Mechanisms for crowdsensing: Motivating Users to Preprocess Data for the Crowdsourcer. *ACM Transactions on Sensor Networks*, 2020, 16(4): 1-24.
23. Yan Ding, Chubo Liu, **周旭**, Zhao Liu, Zhuo Tang. A code-oriented partitioning computation offloading strategy for multiple users and multiple mobile edge computing servers. *IEEE Transactions on Industrial Informatics*, 2019, 16(7): 4800-4810. (SCI 1 区, IF=8.82,一作为共同指导的博士生)

* **已申请/授权相关的国家发明专利：**

1. **周旭**, 罗文晟, 曾一夫, 陈建国, 姜文君, 李肯立, 李克勤. 一种基于社区检索的影响力社区搜索方法和系统. 国家发明专利, 授权专利号: ZL 201910421573.4
2. **周旭**, 苏丹妮, 肖国庆, 曾一夫, 肖正, 李肯立, 李克勤. 一种Top *k*组合空间关键字查询方法和系统. 国家发明专利, 授权专利号: ZL 201910439318.2
3. **周旭**, 刘勇刚, 姜文君, 肖国庆, 罗文晟, 李肯立, 李克勤. 一种实现影响力最大化的初始节点选取方法和系统. 国家发明专利, 授权专利号: ZL 201910448351.1
4. **周旭**, 黄阳, 李肯立, 陈岑, 杨志邦, 曾源远, 罗文晟. 一种基于近似算法的最短路径查询方法和系统. 国家发明专利, 申请号: 2020112898603
5. **周旭**, 桂彧, 肖国庆, 曾一夫, 陈建国, 李肯立, 李克勤. 数据点组查询方法、装置、计算机设备和存储介质. 国家发明专利, 申请号: 2019104605580
6. **周旭**, 翁同峰, 张吉, 余婷, 潘东, 肖国庆, 杨志邦, 陈岑, 李肯立. 基于Quegel分布式图计算系统的Ktruss分解方法. 国家发明专利, 申请号: 202110814231.6
7. **周旭**;苏浩天;顾天乐;张吉;余婷;肖国庆;陈岑;李肯立. 一种不确定的数据交互图中挖掘子图的方法及装置. 国家发明专利, 申请号:202111541216.5