分类号 密级

UDC

学 位 论 文

数据驱动的自适应模式分解算法研究

|  |  |  |  |
| --- | --- | --- | --- |
| 作者姓名： | 岳佳琦 | | |
| 作者学号： |  | | |
| 指导教师： | 王斌 教授 | | |
|  | 东北大学计算机科学与工程学院 | | |
| 申请学位级别： | 硕士 |  |  |
| 学科专业名称： | 计算机科学与技术 | | |
| 论文提交日期： | 2025年5月 | 论文答辩日期： | 2024年6月 |
| 学位授予日期： | 2024年7月 | 答辩委员会主席： |  |
| 评阅人： |  | | |

东 北 大 学

2024年5月

##### A Thesis in Computer Software and Theory

**Research on Data-Driven Adaptive Mode Decomposition Algorithm**

By Yue Jiaqi

Supervisor: Professor Wang Bin

**Northeastern University**

**May 2024**

独创性声明

本人声明，所呈交的学位论文是在导师的指导下完成的。论文中取得的研究成果除加以标注和致谢的地方外，不包含其他人己经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均己在论文中作了明确的说明并表示谢意。

学位论文作者签名：

日 期：

学位论文版权使用授权书

本学位论文作者和指导教师完全了解东北大学有关保留、使用学位论文的规定：即学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人同意东北大学可以将学位论文的全部或部分内容编入有关数据库进行检索、交流。

作者和导师同意网上交流的时间为作者获得学位后：

半年 □ 一年□ 一年半□ 两年□

学位论文作者签名： 导师签名：

签字日期： 签字日期：

# 摘 要

数据库模式分解算法的目标是针对数据库中的某个数据集生成一组用于存储的关系模式，避免数据存储带来的不必要的冗余并利于查询操作的执行。基于函数依赖约束的规范化理论是当前数据库研究领域中最流行且有效的数据库模式分解方法，基于此可以消除函数依赖带来的数据冗余与数据异常。尽管规范化理论已经被数据库社区很好地研究，但将现有的数据集转化为范式（BCNF）仍然是一项非常复杂的任务，特别是云场景中业务逻辑（函数依赖）可能未知的情况下，模式分解是一个NP-hard问题。并且，目前存在的数据库模式分解算法并没有考虑关系模式的变化对工作负载效率的影响，这在实际的应用中会产生非常大的效能影响。因此本文尝试在函数依赖未知的条件下进行模式分解，提出了一种新的数据驱动的自适应模式分解算法。算法通过计算属性之间的依赖关系，挖掘数据之间内在的特征，并通过信息论中的相关理论进行量化，将数据库模式分解问题转换成为组合优化问题进行求解，并分别利用逐层剪枝、和基于图神经网络的方法对该问题进行求解，从而解决了函数依赖未知条件下模式设计困难的问题。针对关系模式对工作负载效率的影响，本文在算法中引入工作负载相关信息，将其作为先验信息纳入模式分解考虑范围，使之能够根据不同用户的需求，自适应的调节模式对空间和时间的增益程度。防止出现分解后的模式对频繁查询出现效率上的负作用。本文提出的算法在通用benchmark数据集上进行了测试，结果表明相比于传统的基于函数依赖与规范化理论的模式分解算法，本文提出的算法可以在函数依赖未知的情况下有效的进行模式分解并且分解后的关系模式对工作负载的效率比传统方法有显著提高。

关键词：数据库；模式分解；数据驱动；信息论；组合优化；图神经网络

### Abstract

The goal of the database schema decomposition algorithm is to generate a set of relational schemas for storing a dataset in a database, avoiding unnecessary redundancy introduced by data storage and facilitating efficient query operations. The normalization theory based on functional dependency constraints is the most popular and effective method for schema decomposition in the current database community, as it can eliminate data redundancy and anomalies caused by functional dependencies. Although normalization theory has been well studied in the database community, transforming existing datasets into Boyce-Codd Normal Form (BCNF) remains a highly complex task, especially when business logic (functional dependencies) in cloud scenarios may be unknown, making schema decomposition an NP-hard problem. Additionally, existing database schema decomposition algorithms do not consider the impact of changes in relational schemas on workload efficiency, which can have significant performance implications in practical applications. Therefore, this paper attempts to perform schema decomposition under the condition of unknown functional dependencies and proposes a novel data-driven adaptive schema decomposition algorithm. The algorithm explores the intrinsic characteristics between attributes by computing their dependency relationships, quantifies them using relevant theories from information theory, transforms the database schema decomposition problem into a combinatorial optimization problem, and solves it using methods such as layer-wise pruning and graph neural network-based approaches. This addresses the difficulty of schema design under unknown functional dependency conditions. To mitigate the impact of relational schemas on workload efficiency, this paper introduces workload-related information into the algorithm, incorporating it as prior information in the schema decomposition process, allowing for adaptive adjustment of the degree of spatial and temporal gains of schemas based on different user requirements. This prevents the occurrence of negative impacts on efficiency due to decomposed schemas on frequent queries. The proposed algorithm is tested on common benchmark datasets, and the results indicate that compared to traditional schema decomposition algorithms based on functional dependencies and normalization theory, the proposed algorithm can effectively perform schema decomposition under unknown functional dependencies and significantly improve workload efficiency compared to traditional methods.

**Key words:** Schema Decomposition; Data-Driven; Information Theory; Combinatorial Optimization; Graph Neural Networks

目录

[第1章 绪 论 1](#_Toc160473968)

[1.1 研究背景及意义 1](#_Toc160473969)

[1.2 国内外研究现状 2](#_Toc160473970)

[1.2.1传统的范式分解 2](#_Toc160473971)

[1.2.2 基于函数依赖挖掘的模式分解 2](#_Toc160473972)

[1.3本文工作内容 3](#_Toc160473973)

[1.4论文组织结构 3](#_Toc160473974)

[1.5本章小结 4](#_Toc160473975)

[第2章 相关理论和技术综述 5](#_Toc160473976)

[2.1数据库范式 5](#_Toc160473977)

[2.2函数依赖挖掘 6](#_Toc160473978)

[2.1模式分解 6](#_Toc160473979)

[2.4信息论基础 8](#_Toc160473980)

[2.5组合优化问题 9](#_Toc160473981)

[2.6本章小结 10](#_Toc160473982)

[第3章 基于工作负载的属性编码研究 11](#_Toc160473983)

[3.1工作负载在模式分解中的意义 11](#_Toc160473984)

[3.2工作负载模板化 12](#_Toc160473985)

[3.2.1工作负载抽样 13](#_Toc160473986)

[3.2.2工作负载模板化 14](#_Toc160473987)

[3.3属性编码 15](#_Toc160473988)

[3.3.1独立热编码 15](#_Toc160473989)

[3.3.3基于图嵌入网络的属性编码 15](#_Toc160473990)

[3.4本章小结 15](#_Toc160473991)

[第4章 基于信息论的目标函数设计 16](#_Toc160473992)

[4.1空间收益建模 16](#_Toc160473993)

[4.2互信息引入与平均互信息 16](#_Toc160473994)

[4.2.1属性间互信息与空间利用率 16](#_Toc160473995)

[4.2.2平均互信息设计 16](#_Toc160473996)

[4.3惩罚因子 16](#_Toc160473997)

[4.4本章小结 16](#_Toc160473998)

[第5章 自适应模式分解算法研究 17](#_Toc160473999)

[5.1算法整体设计 17](#_Toc160474000)

[5.2工作负载处理模块 17](#_Toc160474001)

[5.3数据处理模块 17](#_Toc160474002)

[5.4模式分解算法 17](#_Toc160474003)

[5.5无损性保证 17](#_Toc160474004)

[5.6本章小结 17](#_Toc160474005)

[第6章 实验结果与对比分析 18](#_Toc160474006)

[6.1实验设置 18](#_Toc160474007)

[6.1.1实验环境 18](#_Toc160474008)

[6.1.2数据集介绍 18](#_Toc160474009)

[6.1.3评价指标 18](#_Toc160474010)

[6.2实验结果及其分析 18](#_Toc160474011)

[6.3本章小结 18](#_Toc160474012)

[第7章 总结与展望 19](#_Toc160474013)

[7.1工作总结 19](#_Toc160474014)

[7.2未来展望 19](#_Toc160474015)

[参考文献 20](#_Toc160474016)

[致谢 21](#_Toc160474017)

# 绪 论

## 研究背景及意义

随着信息技术的迅速发展和数据规模的不断膨胀，数据库系统在各行各业的应用中扮演着越来越重要的角色。然而，传统的数据库设计往往面临着诸多挑战，例如数据存储效率低、查询性能不佳、系统可维护性差等问题，这些问题成为数据库管理者和设计者急需解决的难题。因此，数据库模式分解算法应运而生。其主要研究目的是将复杂的数据库设计分解为更小、更易管理的模式，以提高数据库系统的性能、可维护性和扩展性。

模式分解算法旨在通过合理的数据结构组织和优化，提升数据库系统的整体效率和管理水平。它为数据库系统的设计和管理提供了有效的技术手段。通过合理的算法分解数据库模式可以降低系统的复杂度，提高数据库管理的效率和灵活性，从而更好地满足用户的需求。模式分解算法还有助于优化数据库系统的性能。经过模式分解后的数据库关系模式更加合理，可以提高查询效率、降低存储空间占用，从而加速数据访问和处理过程，提升系统的整体性能。此外，对数据库模式分解算法的研究还有助于推动数据库理论的发展。通过对数据库模式分解的深入探索，可以更好地理解数据库中数据之间的关系和依赖，为数据库理论的进一步发展提供重要的理论基础。

除此之外，数据库模式分解算法的研究还对数据库系统的自动化设计和优化具有重要意义。随着人工智能和机器学习等技术的不断发展，自动化数据库设计成为了未来的发展趋势。而模式分解算法作为其中的重要一环，可以帮助数据库系统实现自动化的设计和优化，减少人工干预，提高系统的智能化水平。

然而，随着近年来大数据和云计算的兴起，数据集规模和复杂性呈指数级增长，业务需求也日益多样化、动态化。传统的模式分解算法已经无法满足这种快速变化的需求。因此，基于数据驱动和自适应性的模式分解算法成为研究的焦点。这种算法能够根据关系数据集的特征和工作负载的需求，自适应地调整模式分解的策略，提高系统的整体性能并满足各种各样的用户需求。

## 国内外研究现状

### 1.2.1传统的范式分解

传统的范式分解算法是数据库设计中的一种重要方法，旨在将数据库模式转化为符合规范化范式的形式，以减少数据冗余和提高数据结构的合理性。常见的范式包括第一范式（1NF）、第二范式（2NF）、第三范式（3NF）和Boyce-Codd范式（BCNF）等（上述范式的定义见表1）。传统的范式分解算法主要基于规范化理论通常的做法是根据已知的数据中的函数依赖关系，从第一范式开始，逐步将模式转化为第二范式、第三范式，最终达到BCNF。这些范式的转换过程涉及属性分解、合并和重组，以确保每个关系都满足特定的范式要求。

传统的范式分解算法在数据库设计中发挥着重要作用，尤其是在小型和中型数据库系统中。它们提供了一种系统化的方法来优化数据库结构，减少数据冗余和提高数据的完整性和一致性。然而，范式分解算法也存在一些局限性，如在处理大型数据库和复杂业务逻辑时效率较低，且无法处理部分非规范化的数据设计需求。

### 1.2.2 基于函数依赖挖掘的模式分解

基于函数依赖挖掘的模式分解算法是数据库设计领域的一种重要方法。该算法的核心在于识别数据库中存在的函数依赖关系。函数依赖是指在关系数据库中，一个或多个属性的取值决定另一个属性的取值的关系。通过挖掘函数依赖关系，算法能够深入了解数据库中数据之间的联系和规律，为后续的模式分解奠定基础。这一步骤通常涉及对数据进行分析和探索，以确定属性之间的依赖关系，从而为数据库设计提供重要参考。

一旦函数依赖关系被识别出来，该类模式分解算法便会将这些关系建模为数据库模式的一部分。这意味着将函数依赖关系集成到数据库模式中，以便在后续的模式分解过程中使用。建模过程可以采用各种数据结构，如依赖图或其他形式的数据结构，以有效地表示函数依赖关系之间的联系和影响。通过合理的建模，算法可以更好地理解数据库结构，为接下来的模式分解提供更精确的指导和决策依据。

最后，基于函数依赖挖掘的模式分解算法执行模式分解的步骤，将数据库模式分解为符合特定范式的关系。例如第三范式（3NF）或Boyce-Codd范式（BCNF）。在这个过程中，算法可能会进行属性分解、合并和重组等操作以确保每个关系都满足特定的范式要求，并且保证数据的一致性和完整性。通过模式分解，算法能够优化数据库设计，减少数据冗余和提高数据的一致性，从而提升数据库系统的性能和可维护性。

## 1.3本文工作内容

为了解决模式分解任务面临的上述挑战，规避函数依赖挖掘算法带来的问题，本文提出了一种新颖的数据驱动的自适应模式分解算法算法。该算法为模式分解引入工作负载相关信息，将其作为先验信息纳入模式分解考虑范围，使之能够根据不同工作负载的需求，自适应的调节模式对空间和时间的增益程度。防止出现分解后的模式对频繁查询出现效率上的负作用。并且，该算法使用信息论的相关理论（信息熵、互信息等）直接从数据出发捕获不同属性之间隐藏的关系并量化不同的关系模式对于非规范表的增益程度。最后，该算法借助信息论设计了一个目标函数量化了不同关系模式对空间效益和工作负载效率的好坏程度，将模式分解问题转换成组合优化问题并利用剪枝和基于图神经网络的策略对其进行了有效求解。

本文的贡献点主要总结如下：

1）本文指出现存的函数依赖挖掘算法在模式分解任务上的存在的几个棘手问题。

2）本文设计了一个数据驱动的自适应模式分解算法，成功规避了现存的模式分解算法面临的挑战。

3）实验表明，在函数依赖未知的情况下，本文提出的数据驱动的自适应模式分解算法在时间和空间两个维度上相比现存的基于函数依赖挖掘的模式分解算法算法都有明显的优势。

## 1.4论文组织结构

本文以解决模式分解问题为核心，共分为七个章节。第一章叙述了模式分解在数据库领域中的重要意义、简单介绍了目前的研究现状并简要介绍了本文的工作；第二章详细介绍了本文后续涉及到的相关专业理论与技术；第三章探讨了工作负载在模式分解算法中的意义并对工作负载的模板化方法和基于图嵌入网络的属性编码技术进行了深入研究；第四章探讨了空间收益模型的设计与关系模式目标函数的引入；第五章详细介绍了算法的整体框架；第六章展示了本文的实验结果与分析；第七章对本文的研究进行了总结与展望。

## 1.5本章小结

本章首先阐述了数据库模式分解算法研究的背景和意义，强调了数据库管理系统在信息技术中的核心地位，以及模式分解对数据管理和查询操作的重要性。其次，介绍了传统的范式分解方法及其局限性，以及基于函数依赖挖掘的模式分解算法的相关理论。随后，详细描述了本文工作的内容，即提出一种数据驱动的自适应模式分解算法，以应对现有算法所面临的挑战。最后，概述了本文的组织结构，包括各章节的内容和安排，为读者提供了对全文结构的整体把握。通过本章的内容，读者可以清晰地了解本文研究的背景、目标和组织结构，为后续章节的深入阅读奠定了基础。

# 相关理论和技术综述

## 2.1数据库范式

数据库范式是数据库设计中的一种规范化方法，旨在减少数据冗余、提高数据存储效率和数据完整性。范式通过规定数据库中数据的存储方式，确保数据的每个属性都具有清晰的定义，并且避免数据之间的关系产生不一致性或冗余。主要的数据库范式包括第一范式（1NF）、第二范式（2NF）、第三范式（3NF）以及更高级的范式如BCNF（Boyce-Codd范式）等。

第一范式（1NF）要求数据库表中数据的每个属性都是原子性的，即属性不可再分。例如，如果一个学生的姓名包含了姓和名两部分，则需要将姓名拆分为独立的姓和名两个属性，以满足第一范式的要求。

第二范式（2NF）要求数据库表中的非主属性必须完全依赖于候选键，而不是部分依赖。如果一个表中存在部分依赖的情况，需要将其拆分成多个表，以满足第二范式的要求。其中完全依赖是指一个非主属性完全由候选键中的所有属性决定。换句话说，如果移除候选键中的任何一个属性，非主属性的取值都会发生变化，那么该非主属性就完全依赖于候选键。相反，如果一个非主属性仅依赖于候选键中的一部分属性，而不是全部属性，那么就称为部分依赖。这意味着非主属性的取值部分地依赖于候选键，但还有一部分取值受到其他属性的影响。

第三范式（3NF）要求数据库表中的每个非主属性都不依赖于其他非主属性，即不存在传递依赖关系。如果一个表中存在传递依赖的情况，需要进一步拆分表结构，以满足第三范式的要求。其中，传递依赖指的是非主属性之间存在依赖关系，而不是直接依赖于主键。

更高级的范式如BCNF要求数据库表中的每个属性完全依赖于候选键，且不存在任何对候选键的部分依赖。

数据库范式的优点包括数据存储的紧凑性和数据完整性的提高，通过消除数据冗余和不一致性，提高了数据库系统的性能和可靠性。然而，严格遵循范式可能会导致数据库设计的复杂性增加和查询性能下降。因此，在实际应用中，需要权衡范式的规范性和实用性，根据具体的需求和情况进行灵活选择。

## 2.2函数依赖挖掘

函数依赖挖掘是数据库理论中的一个重要概念，它用于分析数据库中数据之间的关系，特别是用于发现数据的函数依赖关系。函数依赖是指在关系模型中，一个或多个属性的取值可以唯一地决定另一个属性的取值。函数依赖挖掘的目的是发现这些隐藏在数据中的依赖关系，以便进行数据库设计和优化。

在数据库模式分解算法中，函数依赖挖掘扮演着重要的角色。通过挖掘数据库中的函数依赖关系，可以帮助设计者理解数据之间的关系，从而更好地进行数据库模式分解。基于发现的函数依赖，设计者可以将数据库表分解为更小、更简单的子表，使得数据库设计更加模块化和规范化，提高系统的可维护性和性能。

函数依赖挖掘在数据库模式分解中的应用有助于解决冗余数据、提高数据存储效率和数据完整性等问题，从而优化数据库设计和管理。通过合理地利用函数依赖关系，设计者可以构建出更符合业务需求和数据库规范的数据库模式，为系统的发展和维护提供良好的基础。

函数依赖挖掘算法可以被分为四大类:

1）属性格(lattice)：一种基于数据列的函数依赖挖掘算法，通过递归的划分属性来挖掘数据中的函数依赖关系。

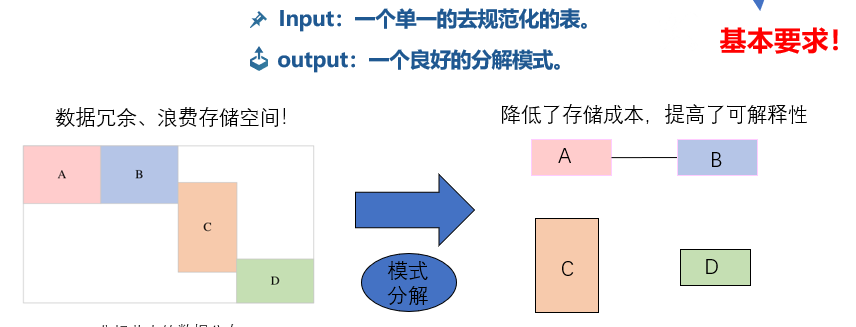
2）一致集(agree sets)：一种基于数据列的函数依赖挖掘算法，该类算法通过计算属性集合的一致集和检验候选函数依赖，逐步发现数据库中的函数依赖关系。

3）混合函数依赖挖掘：将基于行和列的算法相结合，先在列上模拟函数依赖关系，再在行上验证其准确性。

4）基于熵的函数依赖挖掘：将数据转化为二进制向量，并使用互信息度量来发现功能依赖关系，扩展了功能依赖发现的应用范围。（可以详细展开说说）

## 2.1模式分解

数据库模式分解理论是数据库设计和管理领域的重要概念，旨在将复杂的数据库设计分解为更小、更易管理的组件。这个过程有助于实现数据库设计的模块化，提高系统的可维护性、可扩展性和性能。通过功能分解、垂直分解和水平分解等常见做法，可以将数据库设计划分为多个模块，每个模块负责特定的功能或业务逻辑，从而实现更加灵活和高效的数据库管理。此外，通过规范化数据库设计，可以消除数据冗余，提高数据存储效率，支持数据库系统的持续发展和优化。

下图给出模式分解标准定义：

数据库模式分解的意义在于提高数据库系统的可维护性和性能。合理的模式分解可以降低系统复杂度，使得数据库管理更加简单和有效。通过垂直分解和水平分解等方法，可以将数据库设计划分为多个模块，从而实现更好的数据管理和查询性能。此外，模块化的数据库设计也更易于扩展和修改，可以根据系统需求灵活地调整数据库结构，支持系统的持续发展和优化。

总的来说，数据库模式分解理论提供了一种有效的方法来管理和设计复杂的数据库系统。通过合理的模式分解，可以实现数据库设计的模块化，提高系统的可维护性、可扩展性和性能。各种常见做法如功能分解、垂直分解和水平分解等，都有助于将数据库设计划分为多个独立的组件，从而更好地满足系统的需求，支持数据库系统的持续发展和优化。

目前常见的数据库模式分解算法基本都是以数据库范式与函数依赖挖掘为基础设计并实现的。这些算法旨在将给定的宽表分解为符合数据库范式的关系模式，从而减少数据冗余、提高数据存储效率和数据完整性。首先，通过函数依赖挖掘，算法会分析原始数据中的函数依赖关系，然后，基于数据库范式的规范化原则，算法会根据函数依赖关系将宽表分解为符合第一范式（1NF）、第二范式（2NF）、第三范式（3NF）等范式的关系模式。

然而，这种基于函数依赖挖掘和数据库范式的模式分解算法也存在一些缺点。首先，在函数依赖挖掘过程中，算法往往会挖掘出大量的函数依赖关系，然而其中的大部分依赖可能是多余的、冗余的，这使得算法在分解过程中引入了大量的不必要的复杂性和开销。这些冗余的依赖关系不仅增加了计算和存储的负担，也使得最终的数据库模式变得复杂难以理解和维护。其次，这种两阶段分解过程放大了脏数据的影响。在第一阶段的函数依赖挖掘过程中，脏数据或错误数据可能会导致挖掘出错误的依赖关系，进而影响到最终的模式分解结果。这种放大效应使得算法对数据质量要求较高，同时也增加了对数据清洗和预处理的需求，增加了算法的复杂性和实施难度。最后，传统的模式分解算法往往难以平衡空间存储和工作负载效率之间的关系。严格遵循数据库范式可能会导致关系模式的过度分解，产生过多的小型表，查询语句往往需要进行多次表连接操作才能获取所需的结果进而影响了工作负载的执行效率。这种空间存储和工作负载效率之间的权衡往往需要根据具体的应用场景和需求进行平衡，但传统算法往往缺乏有效的机制来实现这种平衡。

这是三张图，论述上面的三个缺点。

## 2.4信息论基础

信息论是研究信息传输、存储和处理的数学理论，其中包含了一些重要的概念，如信息熵、互信息和相对熵等。信息熵是信息论中的核心概念，用于衡量信息的不确定度或平均度量，它描述了一个随机变量的不确定性程度，通常使用香农熵公式进行计算。互信息则是衡量两个随机变量之间相关性的度量，它表示当我们知道一个随机变量的取值时，对另一个随机变量的不确定性的减少程度。相对熵是用来比较两个概率分布之间差异的指标，表示一个分布相对于另一个分布的信息增量或损失程度。

在数据库领域，信息论的应用十分广泛，可以用于数据压缩、数据加密、数据挖掘和数据可视化等方面。首先，信息熵在数据压缩中起着重要作用，通过评估数据的不确定性和复杂性，可以帮助设计更有效的数据压缩算法，提高数据存储和传输的效率。其次，互信息可以用于发现数据之间的关联性，帮助进行数据挖掘和特征选择，从而提取数据中的有用信息。此外，相对熵可用于评估模型的性能和优化数据库查询，帮助数据库系统更加高效地处理数据和查询请求。

信息论在数据库领域的应用不仅有助于提高数据管理和分析的效率，还能够帮助我们更好地理解数据的本质和结构，从而更好地利用数据资源。通过深入研究信息论相关概念，数据库领域可以借助这些理论工具解决实际问题，推动数据库技术的发展和应用。

基于此，本文认为信息论在数据库模式分解算法中能够帮助发现数据之间的潜在依赖关系和模式，从而指导数据库表的分解和优化。通过分析数据的信息熵和互信息，可以揭示数据之间的关联性和重要性，有助于确定哪些属性是紧密相关的，应该放在同一个表中，以及哪些属性之间的关系较弱，可以拆分成不同的表。此外，相对熵可以用来比较不同的数据库模式之间的差异，帮助设计者评估不同方案的优劣并进行决策。因此，信息论在数据库模式分解算法中的应用可以提高数据库设计的规范性和效率。

## 2.5组合优化问题

组合优化问题是一类重要的决策问题，其目标是在给定的一组约束条件下，从所有可能的组合中找到一个最优的解决方案，以最大化或最小化某种指标函数的值。这些问题通常涉及到对一组对象进行选择、排序、分配或组合，以满足特定的需求或优化目标。常见的组合优化问题包括旅行商问题、背包问题、集合覆盖问题等。在求解过程中，通常需要考虑问题的可行性和优化目标，并通过设计有效的算法来搜索解空间并找到最优解或近似最优解。

组合优化问题的求解方法包括精确算法、近似算法和启发式算法等。精确算法试图在有限时间内找到问题的精确最优解，但由于问题的复杂性，这类算法通常只适用于小规模问题。近似算法通过在可接受的时间内找到接近最优解的解决方案来解决问题，而启发式算法则利用问题的特性和经验知识，设计出高效的搜索策略，帮助在大规模问题上找到较好的解决方案。

在数据库模式分解算法中，组合优化问题的研究成果可以为数据库设计提供重要的指导和启示。数据库模式分解涉及将数据库设计分解为更小、更简单的组件，以提高数据库系统的可维护性、可扩展性和性能。由于数据库模式分解本质上也是一个优化问题，因此组合优化问题的求解方法和算法可以被应用于数据库模式分解中，帮助设计者优化数据库结构、提高系统的性能和效率。通过将组合优化问题与数据库模式分解相结合，可以实现对数据库设计的自动化和优化，为数据库系统的设计和管理提供更好的支持和解决方案。

综上，提出本文的大致思路

## 2.6本章小结

本章主要介绍了数据库模式分解、数据库范式、函数依赖挖掘、信息论基础以及组合优化问题等相关理论和技术。数据库模式分解是将复杂的数据库设计分解为更小、更易管理的组件，以提高数据库系统的可维护性、可扩展性和性能。数据库范式是数据库设计中的规范化方法，通过消除数据冗余和不一致性，提高了数据库系统的性能和可靠性。函数依赖挖掘是用于分析数据库中数据之间的关系，特别是发现数据的函数依赖关系，有助于数据库模式分解和优化。目前主流的模式分解算法均是利用函数依赖挖掘算法与数据库范式进行两阶段分解。信息论基础提供了衡量信息不确定性和相关性的工具，可以在数据库领域用于数据压缩、数据挖掘等方面。组合优化问题是一类重要的决策问题，其研究成果可以为数据库模式分解提供重要的指导和启示。综上所述，本章内容有助于读者深入了解数据库模式分解领域的相关理论和技术。

# 第3章 基于工作负载的属性编码研究

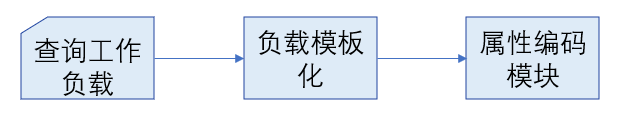
## 3.1工作负载在模式分解中的意义

工作负载是数据库系统面临的实际应用需求和任务，包括用户的查询请求、数据更新操作以及其他数据库操作。而模式分解算法则是为了优化数据库结构，提高数据库系统的性能和可维护性而设计的，它们的目标是将给定的宽表分解为一组占用空间较小的无损的关系模式，从而减少数据冗余、提高数据存储效率和数据完整性。工作负载对于模式分解算法的影响主要体现在以下几个方面：

查询需求： 工作负载反映了数据库系统面临的真实应用场景和用户查询模式。模式分解算法需要充分理解工作负载的特点和查询需求，从而设计出符合实际应用需求的数据库模式。通过分析工作负载中的查询操作和数据访问模式，可以确定数据库模式的划分和优化策略。

查询性能： 工作负载中的查询性能直接受到数据库模式设计的影响。合理的模式设计能够降低查询操作的复杂度和成本，从而提高查询性能和响应速度。模式分解算法通过优化数据库结构，减少数据冗余和提高数据存储效率，有助于降低查询操作的执行时间和资源消耗，从而提高了数据库系统的整体性能。

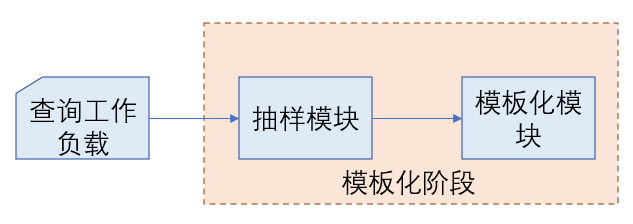
系统可扩展性： 工作负载的特点和变化趋势对数据库系统的可扩展性提出了挑战。一个优秀的模式分解算法需要根据工作负载的特征和变化趋势，灵活调整数据库结构，实现对系统的动态优化和扩展。通过合理的模式设计，可以提高数据库系统的可扩展性，使其能够更好地适应不断变化的工作负载和数据规模。

资源利用率： 工作负载的特点对数据库系统的资源利用率产生重要影响。模式分解算法通过优化数据库结构，合理利用系统资源，提高了数据存储和访问的效率，从而实现了对系统资源的有效利用。通过减少数据冗余和优化数据存储结构，模式分解算法能够降低系统的存储成本和维护成本，提高系统的资源利用率。

工作负载处理流程图

综上，工作负载在模式分解算法中具有重要的参考价值，它反映了数据库系统面临的实际应用需求和任务。是数据库系统运行的核心驱动力。因此，本文提出将工作负载纳入模式分解算法的考量中，使得算法更贴近实际应用场景，让关系数据更好的适应变化的应用场景。本文首先对数据库的工作负载进行了模板化处理，通过提取工作负载中查询的关键特征和模式，将复杂的查询操作转化为简洁的模板形式。随后，针对每个查询模板，对关系中的属性进行编码，具体流程如图。这种编码方式将工作负载中属性之间的查询关系隐含在属性编码中。这样做的目的是为了捕捉工作负载中的重要特征和数据访问模式，并将其纳入后续的模式分解算法的考量范围。在模式分解算法中，我们将这些经过编码的属性信息视为重要的参考依据，结合其他数据库结构的信息，来进行数据库模式的分解和优化。通过将工作负载中蕴含的查询关系纳入模式分解算法的考量范围，可以更全面地理解和分析系统的运行特征，更准确地指导数据库系统的设计和优化。这种做法有助于提高模式分解算法的准确性和效率，使得数据库系统能够更好地适应实际应用场景的需求和变化，提高系统的性能和可维护性。

## 3.2工作负载模板化

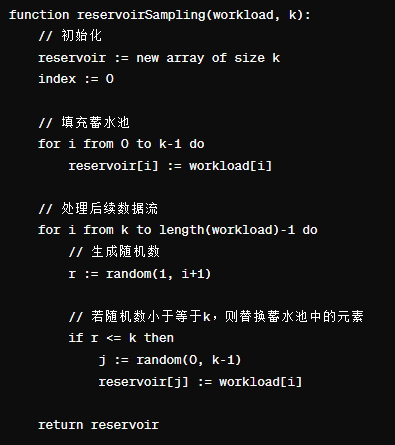
在实际线上数据库环境中，日常查询的数量级通常十分庞大，将所有查询都纳入算法考虑是不现实的，而且大部分查询缺乏典型性，无法代表整体工作负载的特征。因此，对工作负载进行模板化处理具有重要意义。模板化处理通过提取和抽象工作负载中的关键特征和模式，将复杂的查询操作转化为简洁的模板形式。这有助于减少算法分析和处理的复杂度，同时能够更准确地捕捉工作负载的核心特征。通过对工作负载进行模板化，可以过滤掉那些不具代表性的查询，提取出真正具有代表性的查询模式，从而更有效地指导数据库系统的设计、优化和调整。此外，模板化处理还能够降低系统资源的消耗，提高算法处理的效率和性能。因此，对工作负载进行模板化处理是为了更好地理解和分析系统的运行特征，为数据库模式分解算法的设计与优化提供更可靠的依据。模板化流程如下图，首先通

工作负载模板化流程

过抽样方法从大规模的工作负载中获取代表性样本，然后对这些抽样后的查询进行模板化操作。具体内容将在下面小节介绍。

### 3.2.1工作负载抽样

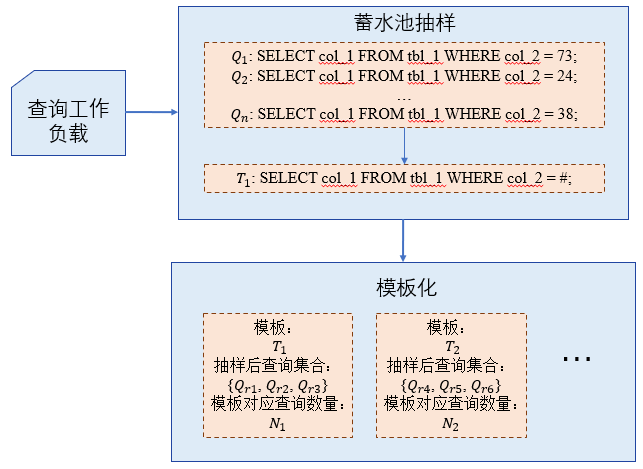
在进行模板化之前需要对工作负载进行抽样的原因在于，数据库尤其是线上的数据库工作负载中通常包含大量的查询或操作，而全面考虑整个工作负载可能会导致计算成本高昂且难以处理。因此，通过对工作负载进行抽样，能够从中提取出具有代表性的样本，以便在后续的模板化操作中更有效地进行分析和处理，从而理解工作负载的特征和模式。

常见的抽样方法包括简单随机抽样、分层抽样、系统抽样和蓄水池抽样等。简单随机抽样是从总体中随机地选取样本，适用于总体分布均匀的情况；分层抽样将总体按照某种特征分成若干层，然后从每一层中分别进行抽样，适用于总体具有明显分层结构的情况；系统抽样是从总体中随机选择一个起始点，然后以一定的间隔从起始点开始抽取样本，适用于无法直接访问总体的情况；而蓄水池抽样是一种用于动态数据流的特殊抽样方法，适用于大规模动态数据流的抽样，且无需知道总体大小。

工作负载蓄水池抽样算法

本文选择了蓄水池抽样对工作负载进行抽样。这是因为工作负载通常是动态的、变化频繁的数据流，且总体规模可能很大，蓄水池抽样能够实现对动态数据流的实时抽样，且无需知道总体大小；其次，蓄水池抽样是一种高效的抽样方法，能够有效地捕捉工作负载的动态特性和变化趋势，保证抽样结果的代表性；最后，蓄水池抽样简单易行，且能够灵活地应用于不同场景下，因此适合于本文的研究需求。对数据库工作负载的蓄水池抽样算法步骤如表：首先初始化一个大小为k的蓄水池，并将前k个数据库查询或操作直接放入蓄水池中。随后，对于后续的每个数据库查询或操作，以一定的概率规则将其加入蓄水池中，并在加入时采用替换机制来保证蓄水池中的元素能够保持一定的随机性。最终，得到的蓄水池即为对工作负载进行抽样的结果。

### 3.2.2工作负载模板化

本文针对抽样后的工作负载进行模板化操作，以服务于模式分解的需求。考虑到模式分解本质是对关系模式的属性进行聚类，本文更加关心属性间的关系而非操作或查询的类型或条件，因此本文将根据操作或查询涉及到的属性对工作负载进行模板化。具体而言，本文将涉及到相同属性的查询视为一类模板进行归纳。这样的模板化操作可以帮助我们更好地理解工作负载中属性间的关联性和重要性，为后续的模式分解提供更加准确和有针对性的信息。

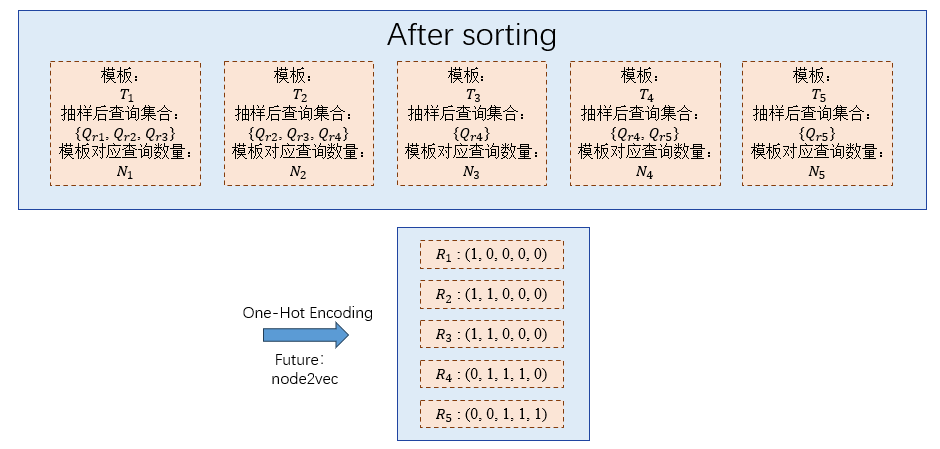
模板化阶段的工作流程图

除了将涉及相同属性的查询视为一个模板外，本文还将对不同模板出现的频率进行统计并排序。这意味着本文不仅仅关注属性间的关联性，还关注不同属性组合的查询在工作负载中出现的次数。通过对模板的频数进行统计排序，我们可以进一步了解工作负载中不同属性组合的重要性和受欢迎程度，从而更好的指导后续的模式分解操作。这种综合考虑属性关联性和模板频数的模板化方法，能够更加全面地把握工作负载的特征，为数据库的模式设计提供更有效的指导和支持。数据库工作负载模板化阶段的工作流程如图。

## 3.3基于工作负载的属性编码及查询相似度设计

工作负载经过模板化后会输出一系列查询模板及其在数据库工作负载中出现的频数。

### 3.3.1位图编码及查询相似度设计

位图编码（Bitmap Encoding）是一种数据压缩和存储技术，通常用于处理大规模数据中的离散属性或标志。在位图编码中，每个属性或标志都被映射到一个位图，其中每个位表示一个可能的取值或状态，位图的长度等于可能的取值个数。

位图编码示例

举例来说，假设有一个包含10个元素的数据集，其中的一个属性是颜色，可能的取值为{"红色", "蓝色", "绿色"}。使用位图编码时，可以创建三个长度为10的位图，分别表示红色、蓝色和绿色属性的存在情况。例如，对于"红色"属性，如果第i个元素具有红色属性，则第i个位图对应的位被设置为1，否则被设置为0。具体到本文中，原始关系数据中的属性对应上述例子中的10个元素，属性在所选的查询模版中是否出现对应上述例子中元素是否有各种不同的颜色，如图所示，例子中包含5个属性并选择了5个查询模板为编码依据，因此创建了5个长度为 5的位图，一次对应不同属性的位图编码。以属性为例，它仅在查询模板中出现过，因此它的位图编码为。其余的属性以此类推，便完成了对所有属性的位图编码。

基于属性的位图编码，本文引入两个属性间的查询相似度，具体定义如公式x。其中，表示属性与属性之间的查询相似度，表示属性的位图编码。公式中分子部分是两属性位图编码的数量积，分母部分是两属性位图编码的模的较大值。分析该公式可知，对于任意的两个属性、,都有。并且，若越接近1，就表明在工作负载中，两属性同时出现在一条数据库操作语句中的可能性越大。为便于理解，本文继续用图中例子举例。通过查询相似度分析图中的属性分别与和的关系可知，对于和，有：

对于和，有：

从查询相似度上来分析，与之间相似度更高，在分解后的关系模式中，更应该出现在同一张关系表中。而回到工作负载中，也可以清晰的看到，属性与同时出现在了频数较高的查询模板中。将它们分到同一张关系表中可以避免在数据库查询时出现大量的表连接操作，有助于提高工作负载效率。因此，本文将查询相似度作为在工作负载层面上，影响模式分解的一部分影响因子。

尽管位图编码和以此为基础设计的查询相似度在属性和工作负载基数相对较少时简洁有效，但当关系数据逐渐复杂，工作负载基数逐渐增大时，位图编码在计算时会消耗大量的内存空间，造成不必要的计算资源的浪费。并且随着属性增多，属性间关系复杂性会指数级的增大，通过相对简单的位图编码对其进行分析也会不够全面。因此，在处理属性数量多，工作负载基数大的关系数据时，本文设计了另一种基于图嵌入基数的属性编码及其对应的属性间查询相似度，见下文。

### 3.3.3图嵌入编码及查询相似度设计

查询图构建（举例并画图），node2vec编码

## 3.4本章小结

本章介绍了数据库工作负载在模式分解中的重要意义，以及基于工作负载的属性间关系量化过程。工作负载是数据库系统面临的实际应用需求和任务，对模式分解算法的设计和优化具有重要影响。本文通过工作负载抽样、模板化、属性编码和查询相似度的设计与计算对属性间关系进行了量化。本文采用蓄水池抽样算法对工作负载进行抽样，以获取代表性样本并通过提取工作负载中的关键特征和模式，将复杂的查询操作转化为简洁的模板形式，这有助于准确捕捉模式分解锁关心的特征。本文将涉及到相同属性的查询视为一个模板进行归纳，并对不同模板出现的频率进行统计。在属性编码阶段，本文提出了基于位图编码和图嵌入编码的两种方法处理不同数量级的关系数据并设计了与其对应的查询相似度。这些方法为模式分解算法提供了重要的参考依据，有助于更好地理解和分析系统的运行特征，指导数据库系统的设计和优化。

# 第4章 基于信息论的目标函数设计

## 4.1空间收益建模

参照ndbc，一堆公式

## 4.2互信息引入与平均互信息

### 4.2.1属性间互信息与空间利用率

从上述公式出发，设计实验，画图

### 4.2.2平均互信息设计

Ndbc

## 4.3惩罚因子

论述空值对平均互信息的影响，画图，引入惩罚因子，实验论证其有效性

## 4.4本章小结

# 第5章 自适应模式分解算法实现

## 5.1算法整体设计

各种模块设计，不同模块负责哪些功能，如何做实验，画图。举一个例子为后续小结说明

## 5.2工作负载处理模块

第4章具体实现，用具体事例进行说明。

## 5.3数据处理模块

第三章具体实现，用具体事例进行说明

5.4模式分解算法

介绍几种组合优化求解方法，介绍本文用到的启发式算法，和图神经网络或遗传算法。

## 5.5无损性保证

两个思路，一个是分的时候带主键，一个是分完堆分配主键

## 5.6本章小结

# 第6章 实验结果与对比分析

## 6.1实验设置

### 6.1.1实验环境

### 6.1.2数据集介绍

### 6.1.3评价指标

## 6.2实验结果及其分析

## 6.3本章小结

# 第7章 总结与展望

## 7.1工作总结

## 7.2未来展望

# 参考文献

# 致谢

在我即将完成研究生学业、撰写毕业论文之际，我要向所有在我研究生生涯中给予过支持、帮助和鼓励的人们致以最诚挚的感谢和敬意。

首先，我要感谢我的导师，王斌教授和杨晓春教授在整个研究生阶段对我的悉心指导和关怀。你们不仅在学术上给予我深刻的启发和指导，还在生活上给予我无微不至的关心和支持。你们的言传身教将成为我人生道路上宝贵的财富。

其次，我要感谢实验室的所有老师和同学们，在我研究过程中提供了许多帮助和支持，与我共同探讨问题，分享经验，使我受益匪浅。

再次，我要感谢我的家人，特别是父母，是你们的支持和理解让我能够专心致志地完成研究生学业。你们对我的关爱和支持是我前进的动力和依靠。

最后，我要感谢所有曾经帮助过我的老师、同学、朋友和亲人们，你们的鼓励、帮助和陪伴让我在求学的道路上不再孤单，是你们让我的研究生生涯充满了温暖和勇气。

感谢你们在我学术和人生道路上的陪伴和支持，我将铭记于心，继续努力前行。同时，我也希望将来能够回馈社会，将所学所得回报给社会，为构建更美好的未来贡献自己的一份力量。

最后，祝愿您们身体健康，工作顺利，生活幸福！

谨以此文，献给所有支持和关爱我的人们。