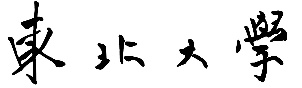


**NORTHEASTERN**

**UNIVERSITY**

硕士学位论文

THESIS FOR MASTER'S DEGREE

|  |  |
| --- | --- |
| 论文题目 |  |
| 作 者 |  |
| 学 号 |  |
| 学 院(部) |  |
| 专 业 |  |
| 指导教师 |  |

年 月

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： | 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

[摘 要 I](#_Toc161047410)

[Abstract III](#_Toc161047411)

[第1章 绪 论 - 1 -](#_Toc161047412)

[1.1 研究背景及意义 - 1 -](#_Toc161047413)

[1.2国内外研究现状 - 2 -](#_Toc161047414)

[1.2.1问题定义 - 2 -](#_Toc161047415)

[1.2.2传统模式分解算法 - 3 -](#_Toc161047416)

[1.2.3 基于函数依赖挖掘的模式分解 - 4 -](#_Toc161047417)

[1.3本文工作内容 - 7 -](#_Toc161047418)

[1.4论文组织结构 - 8 -](#_Toc161047419)

[1.5本章小结 - 8 -](#_Toc161047420)

[第2章 相关技术理论及分析 - 10 -](#_Toc161047421)

[2.1 关系数据库概述 - 10 -](#_Toc161047422)

[2.1.1关系数据库概念 - 10 -](#_Toc161047423)

[2.1.2关系数据库发展历程 - 10 -](#_Toc161047424)

[2.1.3关系数据模型 - 11 -](#_Toc161047425)

[2.1.4关系数据库中基本概念 - 11 -](#_Toc161047426)

[2.4信息论基础 - 12 -](#_Toc161047427)

[2.5组合优化问题 - 14 -](#_Toc161047428)

[2.6本章小结 - 14 -](#_Toc161047429)

[第3章 基于工作负载的属性分析方法 - 16 -](#_Toc161047430)

[3.1工作负载在模式分解中的意义 - 16 -](#_Toc161047431)

[3.2工作负载模板化 - 17 -](#_Toc161047432)

[3.2.1工作负载抽样 - 18 -](#_Toc161047433)

[3.2.2工作负载模板化 - 20 -](#_Toc161047434)

[3.3基于工作负载的属性编码及查询相似度设计 - 20 -](#_Toc161047435)

[3.3.1位图编码及查询相似度设计 - 20 -](#_Toc161047436)

[3.3.3图嵌入编码及查询相似度设计 - 22 -](#_Toc161047437)

[3.4 有效性实验 - 25 -](#_Toc161047438)

[3.5本章小结 - 27 -](#_Toc161047439)

[第4章 基于信息论的目标函数设计 - 29 -](#_Toc161047440)

[4.1空间收益建模 - 29 -](#_Toc161047441)

[4.2互信息引入与平均互信息 - 30 -](#_Toc161047442)

[4.2.1属性间互信息与空间利用率 - 30 -](#_Toc161047443)

[4.2.2平均互信息设计 - 31 -](#_Toc161047444)

[4.3惩罚因子 - 32 -](#_Toc161047445)

[4.4本章小结 - 32 -](#_Toc161047446)

[第5章 自适应模式分解算法框架 - 33 -](#_Toc161047447)

[5.1算法整体设计 - 33 -](#_Toc161047448)

[5.2工作负载处理模块 - 33 -](#_Toc161047449)

[5.3数据处理模块 - 33 -](#_Toc161047450)

[5.5无损性保证 - 33 -](#_Toc161047451)

[5.6本章小结 - 33 -](#_Toc161047452)

[第6章 实验结果与对比分析 - 34 -](#_Toc161047453)

[6.1实验设置 - 34 -](#_Toc161047454)

[6.1.1实验环境 - 34 -](#_Toc161047455)

[6.1.2数据集介绍 - 34 -](#_Toc161047456)

[6.1.3评价指标 - 34 -](#_Toc161047457)

[6.2实验结果及其分析 - 34 -](#_Toc161047458)

[6.3本章小结 - 34 -](#_Toc161047459)

[第7章 总结与展望 - 35 -](#_Toc161047460)

[7.1工作总结 - 35 -](#_Toc161047461)

[7.2未来展望 - 35 -](#_Toc161047462)

[参考文献 - 36 -](#_Toc161047463)

[致谢 - 37 -](#_Toc161047464)

# 绪 论

本章论述了数据库模式分解算法的研究背景及意义，介绍了目前模式分解算法的国内外研究现状并提出了当前研究面临缺点与挑战。并提出了本文对于该算法的贡献。1.1小节中详细论述了模式分解算法的研究背景与意义；1.2小节从问题定义、传统的模式分解算法和目前流行的基于两阶段的算法上介绍了对本文课题的研究现状并提出目前研究未解决的一些缺点问题；1.3小节介绍了本文工作的主要内容与贡献；1.4小节对本文的框架进行了简单梳理。

## 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。

## 1.2国内外研究现状

### 1.2.1问题定义

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。

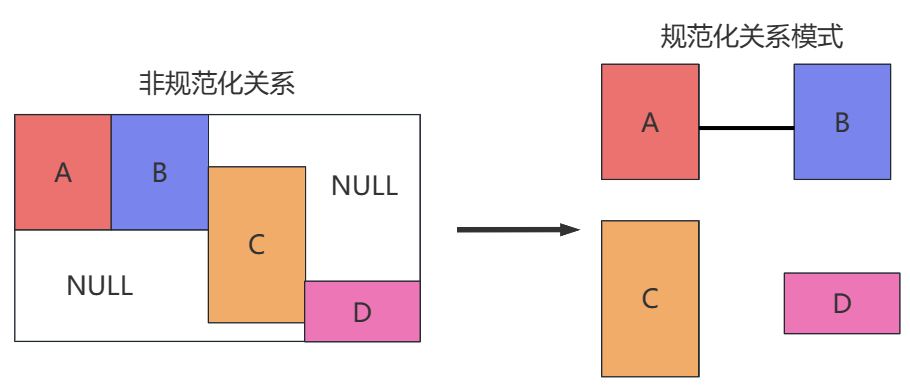
下图给出模式分解标准定义：

图1.1 模式分解问题定义

简单来说，数据库模式分解是指将给定的宽表（非规范化的关系）分解成一组较小的无损的关系模式（规范化的关系模式）的过程，以提高数据库系统的性能和可维护性。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。如图1所示，非规范化关系中存在大量的空值，浪费了大量的存储空间。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。

### 1.2.2传统模式分解算法

基于范式的模式分解算法是一种在关系数据库设计中常用的算法，旨在将一个不符合范式（Normalization）的关系模式分解为符合范式的多个关系模式。范式是一组规范化的设计原则，它有助于消除数据中的冗余、依赖和插入、更新、删除异常等问题，提高数据库的结构化程度和数据的质量。其过程是在关系模式中属性间函数依赖已知的前提下，按照数据库范式的约束等级，逐级判断当前关系模式是否符合范式规范要求，若不符合，则按照范式的要求通过分解将其转换为符合的规范化范式。

其中，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。如下图，是一个将非规范化表分解成满足第一范式的模式例子。

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。如下图，是一个将非规范化表分解成满足第二范式的模式例子。

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。如下图，是一个将非规范化表分解成满足第三范式的模式例子。

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，传统的基于范式的模式分解算法有需要一个大前提，那就是关系中属性之间的函数依赖关系已知，然而随着时代的发展，大数据时代和云时代的到来，越来越多的庞大数据集并不能很容易的得到函数依赖，导致基于范式的模式分解束手无策。因此，业内提出了基于函数依赖挖掘的模式分解算法，用来处理类似的问题。

### 1.2.3 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。函数依赖是关系模型中描述属性之间关系的重要概念，它描述了一个属性或属性集合对于其他属性的决定作用。函数依赖挖掘的目标是通过分析关系实例中的数据，识别出其中存在的函数依赖关系[13]。这有助于了解关系实例中数据的结构、关联性和语义含义。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。

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。

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。这些算法均基于一个两阶段的过程。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。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。显然，这种两阶段过程解决了传统的模式分解算法在函数依赖未知情况下的困境，函数依赖挖掘算法的提出能够帮助我们在函数依赖未知的情况下先挖掘再分解。

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。本文对其进行分析，并通过实验暴露了这些缺点。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。表1展示了目前的几种算法在对tpcc数据集进行模式分解实验时，中间过程产生的函数依赖。从表中可以直观的看到，目前常见的两阶段模式分解算法在函数依赖挖掘阶段会挖掘出大量的函数依赖，远远超过了后续模式分解的需求，正常基于范式的模式分解也不可能按照如此大量的函数依赖进行分解。目前的这些算法的做法是对所有挖掘出的函数依赖进行置信度估计，并根据属性个数挑选置信度最高的几种函数依赖进行挖掘。尽管这种方法可以解决函数依赖个数过多的问题，但这样操作导致剩余挖掘出的函数依赖完全没有应用到后续的模式分解过程。相当于，在第一阶段函数依赖挖掘时，绝大部分的计算资源都被浪费掉了。这对算法的运行效率会产生极大的影响。

表格1.1 函数依赖个数统计表

|  |  |
| --- | --- |
| 方法类别 | 挖掘函数依赖个数 |
| Hyfd | 29834 |
| Pyro | 65947 |
| Fastfds | 324477 |
| Tane | 24767 |

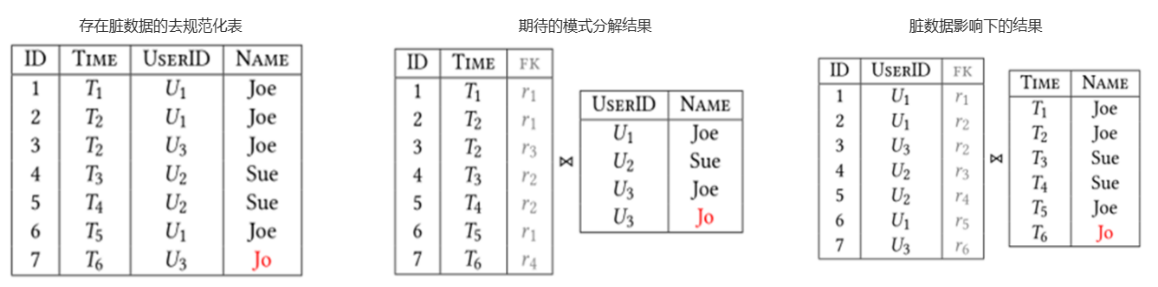
其次，这种两阶段分解过程放大了脏数据的影响。在第一阶段的函数依赖挖掘过程中，脏数据或错误数据可能会导致挖掘出错误的依赖关系，进而影响到最终的模式分解结果。这种放大效应使得算法对数据质量要求较高，同时也增加了对数据清洗和预处理的需求，增加了算法的复杂性和实施难度。如图1.2

图1.2 脏数据对模式分解的影响

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，但目前的数据库模式分解算法往往缺乏有效的机制来实现这种平衡。

## 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并进行了实现。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）本文提出并实现了一种基于工作负载的属性分析方法，并针对关系规模，分别设计了基于位编码和基于图嵌入编码的属性编码方法，能够有效的从工作负载的角度捕获并量化属性间的关系。

2）本文提出并实现了一种基于信息论的属性分析方法，能够从空间利用率的角度分析属性间的关系。能够在保证空间收益的前提下，规避脏数据对结果造成的影响，与传统的函数依赖挖掘算法相比，本文提出的方法节约了大量的计算资源并能取得更好的空间收益。

3）本文提出并实现了一种数据驱动的自适应模式分解方法，该方法综合考量了关系模式分解前后的工作负载效率和空间利用率，将模式分解问题转化成组合优化问题进行求解。并能根据用户的实际需求自适应的调节对两个方面的匹配程度。

4）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能够有效的适应不同业务需求下的模式分解需求。并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

图表 1论文结构图

本文以解决模式分解问题为核心，共分为七个章节，结构如下图。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，以及目前常见的基于函数依赖挖掘的模式分解算法的相关理论并通过实验描述了这些算法的缺点。随后，本章详细描述了本文工作的内容与贡献，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，本章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。

# 相关技术理论及分析

本章介绍了与数据库模式分解算法相关的基础理论知识和后续本文中使用活涉及到的各种理论与技术，便于读者对后续章节的阅读。其中，2.1小节详细介绍了有关关系型数据库的理论和概念；2.2小节详细介绍了关于模式分解问题的定义与使用到的各种概念；2.3小节介绍了信息论的相关理论，在后续的第四章中应用；2.4小节介绍了图嵌入技术，在后续的第三章节中应用；2.5小节介绍了组合优化问题的相关概述，为算法的设计打下基础。

## 2.1 关系数据库概述

### 2.1.1关系数据库概念

关系数据库是一种使用关系模型来组织和管理数据的数据库系统。在关系数据库中，数据以表格的形式进行存储，每个表格都由行和列组成，行表示记录，列表示属性。每个表格都有一个唯一的名称，并且每个属性都有一个特定的数据类型。映射到现实世界中来说，每一个表都可以看做一个实体，表中的每一列对应实体的各种属性，关系模式就是指这些属性间的关系。

### 2.1.2关系数据库发展历程

关系数据库的发展可以追溯到上世纪70年代初期，当时IBM的科学家Edgar F. Codd提出了关系数据库模型的理论基础。1970年，Codd在IBM的一篇论文中首次提出了关系数据库的概念，该论文发表在《Communications of the ACM》杂志上，题为《关于关系数据库的提议》（"A Relational Model of Data for Large Shared Data Banks"）。这篇论文奠定了关系数据库理论基础，并成为关系数据库发展的里程碑之一。

在此基础上，IBM等公司陆续推出了基于这一模型的商业化数据库产品，如IBM的DB2、Oracle的Oracle Database等。这些数据库系统在业界得到了广泛应用，推动了关系数据库技术的发展。

1990年代后期以及21世纪初期，开源关系数据库MySQL、PostgreSQL等也相继涌现。这些开源数据库系统提供了免费、灵活的数据库解决方案，受到了开发者和企业的欢迎，推动了关系数据库技术的普及和应用。

此外，随着互联网的发展和大数据技术的兴起，关系数据库面临着新的挑战和机遇。NoSQL（Not Only SQL）数据库等新型数据库技术逐渐崭露头角，针对大规模、高并发、非结构化数据等特点提出了新的解决方案。同时，关系数据库厂商也不断推出新的功能和服务，以满足用户对于安全性、可扩展性、性能等方面的需求。

### 2.1.3关系数据模型

数据模型包括多种形式，其中关系模型、实体关系模型以及面向对象数据模型等是其中重要的代表。在这些模型中，关系模型作为数据库设计的基础，其构成要素包括关系数据结构、关系操作集合以及关系完整性约束。

1.关系数据结构

关系数据结构是关系数据库中的核心概念之一，用于组织和存储数据。具体来说，在实际生活中的各种物品（实体）均可以用关系来描述，而其抽象后得到的逻辑结构可用二维表来表示。

2.关系操作集合

关系操作集合是在关系数据库中对数据进行操作的一系列操作的集合。这些操作可以对数据进行查询、插入、更新和删除等操作，以实现对数据库的管理和维护。

3.关系完整性约束

关系完整性约束是关系数据库中用于保持数据一致性和完整性的规则和限制。这些约束条件确保了数据库中的数据满足特定的规范，防止了数据的不合法或不一致的情况发生。

关系模型通过对数据进行结构化的表格化表示，使得数据之间的关系能够以清晰、简洁的方式呈现。关系操作集合定义了对数据进行操作的一系列操作，如插入、更新、删除等，这些操作使得用户能够有效地对数据库进行管理和维护。同时，关系模型还通过一系列的完整性约束来确保数据的一致性和完整性，例如主键约束、外键约束等，这些约束条件对数据库的正确性和可靠性至关重要。

### 2.1.4关系数据库中基本概念

在关系数据库中，有一些基本概念是非常重要的，它们构成了关系数据库的核心。以下是关系数据库中一些重要的基本概念。

1. 关系：关系是关系数据库中最基本的概念之一，它类似于表格，用于存储数据。每个关系由多行和多列组成，其中每一行代表一个记录，每一列代表一个属性。
2. 实体：存在于现实世界中并能够相互区分的事物称之为实体。
3. 属性：关系中的每一列就是关系的属性，其名就是属性名。
4. 域：具有相同属性的一类数据的集合。
5. 关系模式：关系的描述称为关系模式。可以形式化的表示为, 其中为关系名，为关系的属性集合，为属性集合中属性的域，为属性间的函数依赖关系集合。

## 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。它最初由克劳德·香农（Claude Shannon）于1948年提出，主要用于研究通信系统中信息的传输和编码方式。信息论的核心思想是通过测量信息的不确定性和随机性来量化信息的内容和传输效率。

其中包含了一些重要的概念，如信息量、信息熵、互信息和相对熵等。下面给出一些重要概念的定义。

信息熵是信息论中的一个重要概念，用来表示一个随机变量的不确定性或信息量的期望值。信息熵的定义如下：假设一个离散随机变量取值于有限个事件，其概率分布为，其中。则随机变量的信息熵定义为：

其中，为自然对数。信息熵衡量了随机变量的平均信息量，即每个事件发生所提供的平均信息量的期望值。信息熵越高，表示随机变量的不确定性越大，每个事件提供的信息量越多；反之，信息熵越低，表示随机变量的不确定性越小，每个事件提供的信息量越少。

互信息是信息论中用来衡量两个随机变量之间相关性的概念。互信息表示的是一个随机变量中包含的关于另一个随机变量的信息量。具体来说，对于两个离散随机变量和，其互信息定义如下：

其中，是和同时取值和的联合概率分布，和分别是和的边缘概率分布。互信息衡量了两个随机变量之间的相互依赖程度。当和独立时，互信息为零；当和相互关联时，互信息为正值，且互信息越大，关联程度越强。

相对熵，也称为散度是信息论中的一种度量两个概率分布之间差异的方法。它用来衡量两个概率分布之间的相似程度或距离。具体来说，对于随机变量的两个概率分布函数和，相对熵定义为：

相对熵的值越大，表示两个概率分布之间的差异越大；相反，相对熵的值越小，则表示两个概率分布之间的差异越小。

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。

本文认为，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。

## 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。

# 基于工作负载的属性分析方法

数据库系统的功能不仅仅限于数据存储，它还负责数据的高效查询和检索。因此，在对复杂的关系数据集进行模式分解时，除了需要优化空间存储效率，还需要进一步考虑对工作负载的查询效率，以保证数据库的性能和响应速度。然而，相关工作中介绍的传统的基于范式的模式分解算法和目前常见的两阶段模式分解算法均忽视了对相关数据集工作负载的统计分析，导致分解后的关系模式在应用时难以达到理想的工作效率。因此本文提出了基于工作负载的属性分析方法，提高了模式分解算法对工作负载的匹配度。本章在3.1节分析了工作负载与模式分解算法的关系，在3.2、3.3小节中介绍了基于工作负载的属性分析方法，包括工作负载抽样、模板化、属性编码方法等。3.4节从分解后的对工作负载的查询效率上证明了该方法的有效性。最后3.5小节中总结了本章的内容。

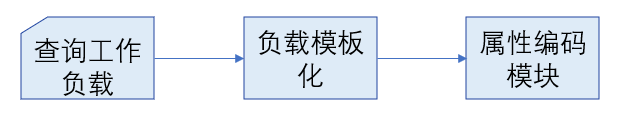
## 3.1工作负载在模式分解中的意义

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。

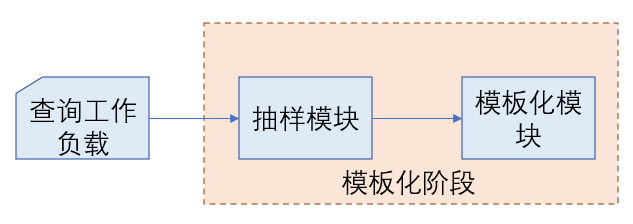
工作负载处理流程图

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。

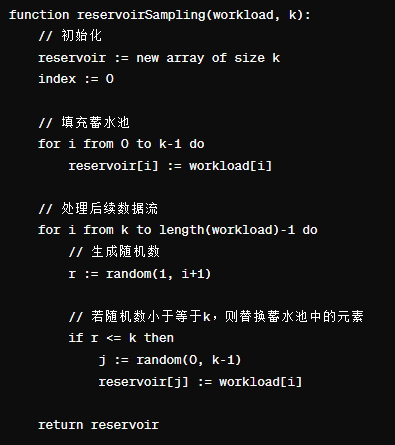
## 3.2工作负载模板化

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从查询的角度进行模板化操作。具体内容将将在后续小节介绍。

### 3.2.1工作负载抽样

本文首先对数据库工作负载进行了抽样，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。

工作负载蓄水池抽样算法

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。

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

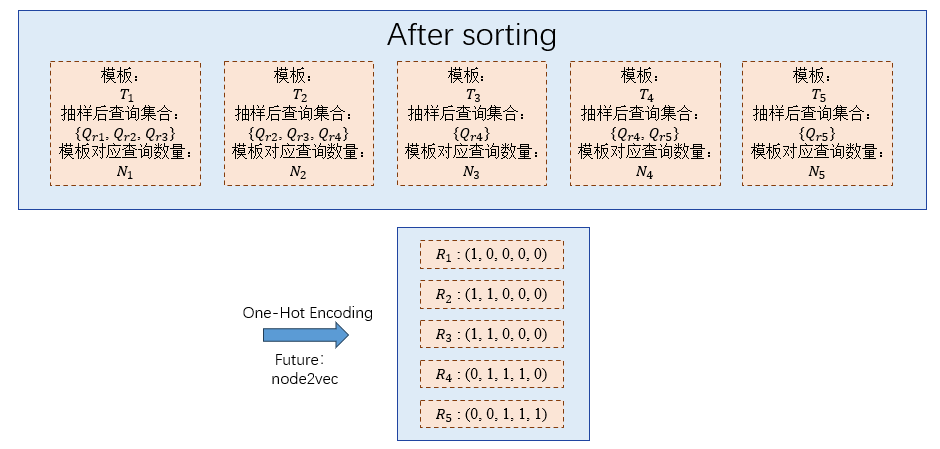
本文针对抽样后的工作负载进行模板化操作，以服务于模式分解的需求。考虑到模式分解本质是对关系模式的属性进行聚类，本文更加关心属性间的关系而非操作或查询的类型或条件，因此本文将根据操作或查询涉及到的属性对工作负载进行模板化。具体而言，本文将涉及到相同属性的查询视为一类模板进行归纳。这样的模板化操作可以帮助我们更好地理解工作负载中属性间的关联性和重要性，为后续的模式分解提供更加准确和有针对性的信息。

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

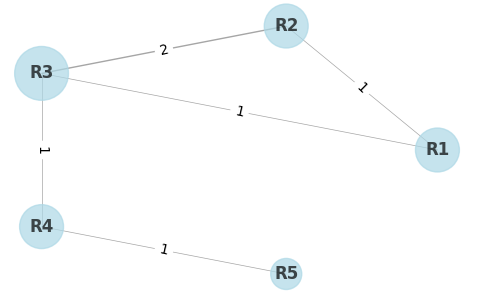
图嵌入算法是一种重要的技术，旨在将图中的节点映射到低维向量空间，以便在保持节点之间结构关系的同时，将节点表示为连续的低维向量。这种表示可以帮助我们理解和分析图中的节点相似性、关联性和特征。图嵌入算法主要分为基于因子分解、基于随机游走和基于深度学习的方法。

基于因子分解的方法通常通过对图的邻接矩阵进行分解来学习节点的向量表示。这些方法包括奇异值分解（SVD）和主成分分析（PCA）。尽管这些方法简单有效，但通常只能处理小规模的图数据，并且在保留图的结构信息方面存在一定的局限性。

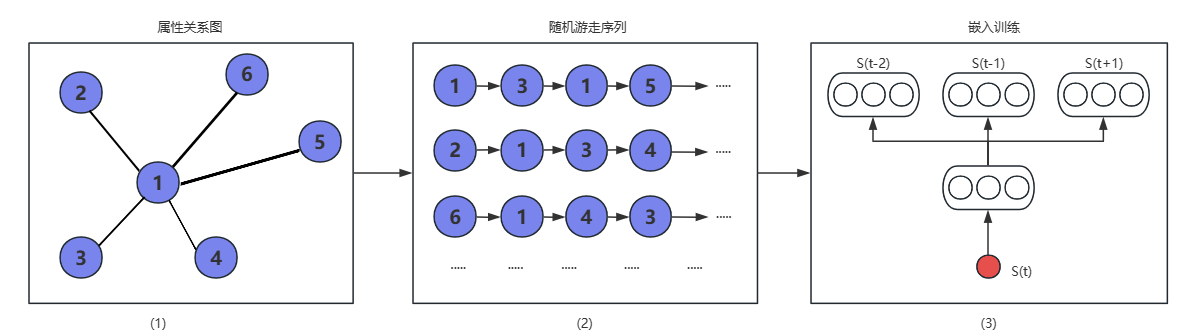
基于随机游走的方法通过模拟节点在图上的随机游走路径来学习节点的向量表示。其中DeepWalk和Node2Vec是最常用的算法，它们通过随机游走生成节点序列，并利用这些序列来训练模型，从而学习节点的向量表示。Node2Vec算法在DeepWalk的基础上引入了参数p和q，用于控制游走策略，以更好地平衡保留局部和全局结构之间的权衡关系。

基于深度学习的方法利用神经网络模型学习节点的向量表示，其中常见的方法包括图卷积网络（GCN）、图注意力网络（GAT）和GraphSAGE等。这些方法通过将图的结构信息编码为神经网络的层次结构，有效地学习节点的向量表示。相较于传统方法，基于深度学习的方法通常能处理更大规模的图数据，并具有更高的灵活性和表达能力。

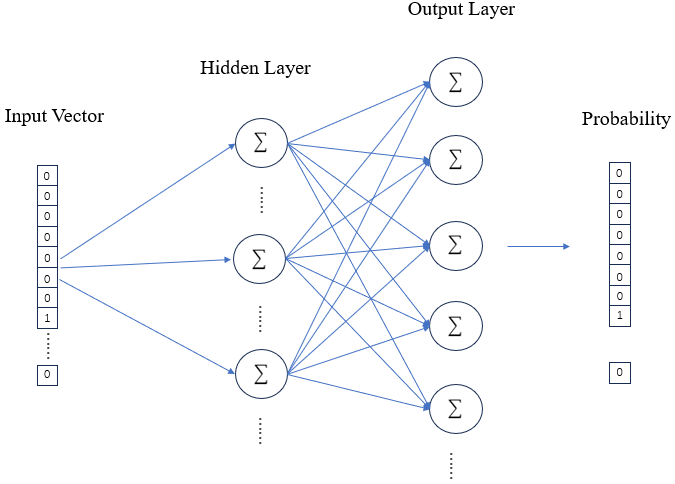
本文使用node2vec嵌入方法实现对出现在工作负载中的关系数据集中的各种属性进行嵌入式编码。首先，本文以工作负载为基础，构建了一张属性关系图，该图能够反应属性之间基于查询的关系。在该属性关系图中，节点代表关系数据集中的属性，边则表示这两个属性是否在查询模板中同时出现。边的权值表示了这两个属性在查询模板中同时出现的频率。



以图中为例，以该工作负载为基础构建的属性关系图如下图。以属性R2与R3

属性关系图为例，在经过抽样模板后的工作负载模板中，与同时在模板与中出现，因此与之间由一条权重为2的边相连。这种图的构建方法将工作负载抽象成一个无向带权图，将工作负载中包含的信息转移到图中。再通过node2vec算法将图中节点嵌入到向量中。

传统的node2vec算法对图节点的嵌入过程如下图，包括随机游走和嵌入学习两个阶段。在随机游走阶段中，算法会从图中的每个节点开始根据节点之间的连接关系执行一系列随机游走操作，并记录下每次游走经过的序列。为了在游走过程中兼顾深度优先和广度优先的特性，node2vec还引入了参数和来调整随机游走的转移概率。参数控制了向前移动的概率，而参数q则控制了向后移动的概率。通过这两个参数，算法可以在深度优先和广度优先之间取得平衡，以更好地探索图结构。接着，node2vec算法将生成的随机游走序列作为输入，利用word2vec等模型学习节点的嵌入表示。word2vec模型根据节点序列中的上下文信息来学习节点的嵌入向量，从而捕捉到节点之间的语义信息。通过这种方式，node2vec算法可以将每个节点表示为一个低维的向量。

本文选用了经典的skip-gram网络模型学习嵌入式向量。skip-gram网络是一种用于学习词嵌入（word embeddings）的神经网络架构。它在自然语言处理（NLP）领域中被广泛用于捕捉词汇之间的语义和句法关系。skip-gram网络的目标是输入一个目标词汇来预测目标词的上下文（即周围的词）。这种模型适用于文本语料库，因为它能够捕捉到词汇之间的复杂关系。在Skip-gram模型中，每个词都被表示为一个高维空间中的向量，这些向量被称为词嵌入。这些嵌入向量是通过训练神经网络来学习的，使得语义上相似或在文本中经常一起出现的词在向量空间中彼此靠近。在本文的应用中，将在属性关系图中生成的每一条随机游走序列视为自然语言处理中的每一句话；将随机游走序列中的各个序列号视为构成语句的每个单词。而随机游走序列又是从本文基于工作负载构造的属性关系图中生成的，因此本文中skip-gram网络模型可以通过在训练过程中预测某个节点的临近节点的概率而不断进行调整，进而学习每个节点的词嵌入向量。

具体的学习过程如图所示，skip-gram 网络模型的输入通常是一个节点的 one-hot 编码表示，其中一个维度对应于一个节点。然后，该 one-hot 编码被传递到一个浅层的神经网络结构中，该结构通常包括输入层、隐藏层和输出层。在神经网络中，输入层的节点表示中心节点，而输出层的节点表示与中心节点具有关联的邻近节点。通过隐藏层中的权重矩阵，skip-gram 网络将输入节点表示映射到嵌入向量空间中得到它的嵌入向量。通过训练神经网络，skip-gram 网络试图最大化给定中心节点时，与之关联的邻近节点的概率。具体来说，skip-gram 网络利用了 softmax 函数来计算邻近节点的概率分布，并通过最大化这个概率分布来优化节点嵌入的表示。

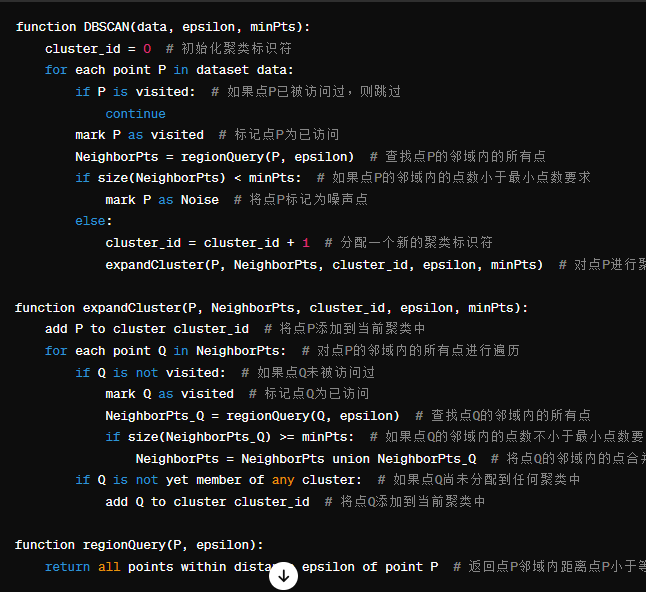
需要注意的是，传统的 node2vec处理的是无权图，而本文生成的属性关系图为有权图，因此本文在传统的node2vec算法基础上对第一阶段的随机游走策略进行了改进，使之根据节点之间边的权重值来对随机游走的概率进行调整。从本文需求的角度，边权越大，选择该边游走的可能性也应该越大，因此本文设计了带权随机游走策略如下，其中随机游走的概率与边的权重成正比。

与基于位图的属性编码类似，对于基于属性的图嵌入编码，本文也引入了衡量两个属性间的查询相似度，具体定义如公式如下

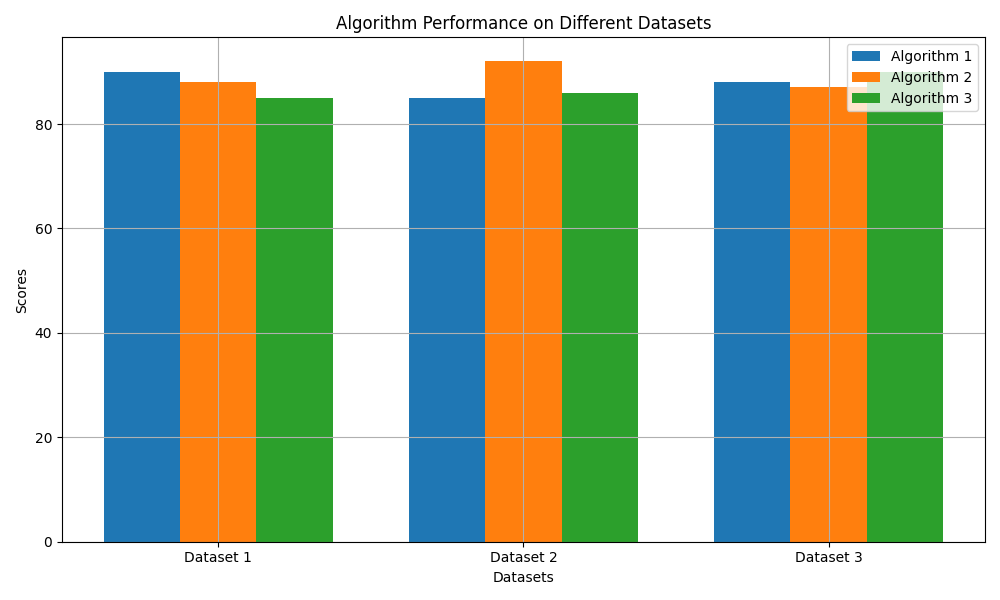
基于图嵌入属性编码的查询相似度利用了余弦相似度，与基于位编码的查询相似度类似，对于任意的两个属性、,都有。并且，若越接近1，就表明在本文构建的属性关系图中两属性代表的节点在我们定义的特征上越相近，即两属性同时出现在一条数据库操作语句中的可能性越大。

## 3.4 有效性实验

为了验证本文提出的基于工作负载的属性分析方法的有效性，本文依据按照上文提出的方法分析计算出的查询相似度，对不同关系数据集的工作负载进行处理，并按照计算得到的查询相似度对其涉及的属性进行聚类。本小节将属性聚类视作一个简单的模式分解过程，将聚类结果的不同类别视为将属性划分到了不同的表中，并统计聚类结果的单表查询覆盖率。单表查询覆盖率是指工作负载中，不需要进行表连接的查询语句数量与工作负载查询语句总数的比值。因为在数据库查询覆盖率。单表查询覆盖率是指工作负载中，不需要进行表连接的查询语句数量与工作负载查询语句总数的比值。因为在数据库查询过程中，表连接操作往往会带来额外的计算开销和数据传输成本。因此，单表查询覆盖率越高，数据库的查

询性能和效率越高。聚类算法如下表，

需要注意的是，本小节中设计的实验仅仅是为了验证本章提出的基于工作负载的属性分析方法对工作负载中蕴含的属性之间关系的捕获效果，以提升工作负载的查询效率。因此，并未考虑在模式分解过程中添加主外键以确保分解的无损性。实际上，针对本小节实验中的指标——单表查询覆盖率而言，是否保证无损性并不会对其产生显著的影响，反而会大大增加算法与实验的复杂性。本文提出的系统的保证无损性的模式分解算法在后续的第五章中有详细的介绍。

本节将上述算法结果与其他常见的基于函数依赖挖掘算法的模式分解算法结果的单表查询覆盖率进行了对比，如下图：

这里有一个柱状图

从图中可知，本节通过对比基于工作负载的属性分析方法与其他常见的基于函数依赖挖掘算法的模式分解算法的结果，可以很明显的看出基于工作负载的属性分析方法指导的聚类算法的单表查询覆盖率对比其他算法均有显著提高。这表明该方法能够有效地捕获属性之间基于工作负载的关系，并将其量化应用于后续阶段，从而提高数据库查询效率。

## 3.5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。在最后一节中，本章设计了对比实验，证明了本章提出的基于工作负载的属性分析方法可以有效的从工作负载中提取出属性之间的关系，并将这种关系进行量化应用与后续阶段从而提高数据库的查询效率。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。

# 基于信息论的目标函数设计

本章针对传统的基于范式的模式分解算法无法在属性间函数依赖未知的情况下有效对关系数据进行分解的问题，和目前常见的基于函数依赖挖掘算法的两阶段模式分解过程造成的计算资源浪费和对脏数据敏感的问题，提出了基于信息论的属性分析方法。该方法可以在确保模式分解后空间增益的前提下分析关系数据中属性间的关系，避免大量的无用计算，并且可以尽可能避免脏数据对模式分解结果造成影响，提高算法的鲁棒性。4.1小节中本文从存储空间利用率的角度对模式分解前后的空间收益进行建模，并从空间收益的角度上将数据库模式分解问题转换成组合优化问题。4.2小节中本文讨论了互信息与空间利用率之间的关系，并引入平均互信息量化4.1小节中的空间收益模型。4.3小节中，本文论述空值对平均互信息的影响并引入惩罚因子避免空值对平均互信息结果造成影响。4.4节从分解后的对空间增益率证明了该方法的有效性。最后在4.5小节中总结了本章的内容。

## 4.1空间收益建模

数据库模式分解算法的目的之一是减少关系数据中的数据冗余，提高存储空间的利用率。然而，目前常见的基于两阶段过程的模式分解算法并没有对模式分解前后存储空间利用率进行量化，而只是通过函数依赖的挖掘，间接实现了提高空间利用率的目的。然而这种做法有很大的缺陷，从关系数据出发进行函数依赖挖掘的过程易受脏数据影响，并会不可避免的产生大量计算资源的浪费，绪论中已通过实验论述。因此本章提出由数据出发，直接指导算法对关系数据进行分解，这种方法可以杜绝上述提到的缺陷。

本章旨在通过模式分解方法优化数据库的空间利用率，以提高数据库系统的性能和效率。在这一过程中，对空间收益进行建模具有重要意义。下面是对模式分解前后空间收益进行建模的过程和相关定义。

给定一个关系实例，其属性集合为。数据库模式分解算法尝试将R用一个更小的无损的关系模式组合来替代，其中构成关系的属性集为，且，（无损性保证），如果分解后的关系实例所占空间小于分解前的关系实例R，则算法生成的模式S是有效的。

本文将寻找关系实例的最优模式定义为最优化问题：

其中：

上述定义中， 表示关系实例使用的单元格数（单属性列单行元素占用个单元格），其中，表示实例关系的属性个数，表示实例关系在属性集上的基数，即关系在属性集上的投影的行数； 表示模式使用的单元格数，其中和分别表示模式的第个关系的属性个数和其在属性集上投影的总行数。 表示分解模式 所带来的空间增益。式(4.1a)从空间增益的角度上将最优模式定义为导致最大空间增益的模式S。

## 4.2互信息引入与平均互信息

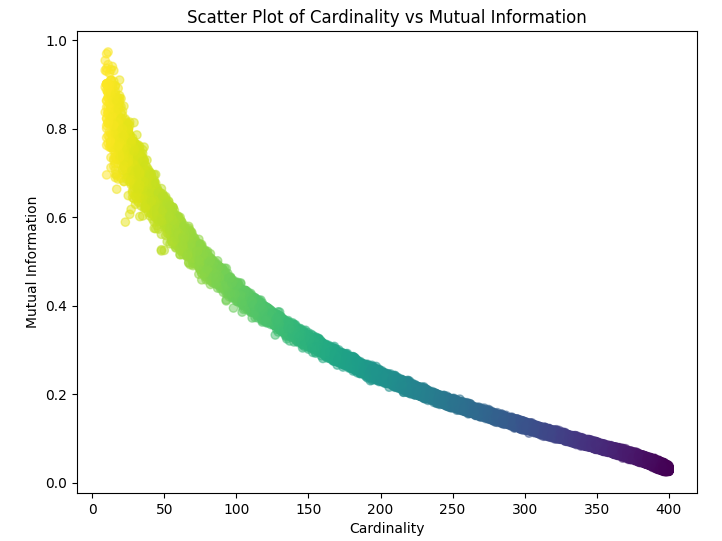
### 4.2.1属性间互信息与空间利用率

模式设计的目标是将一个宽表R分解为一组占用空间较小的无损的关系模式，实现对数据存储空间的优化。分析式（3.1），在给定实例关系的前提下， 是一个常数，因此最大化 等价于：

由式(4.2)可知，影响分解前后关系模式空间收益的主要因素是分解后关系模式中各关系在各自属性集上投影的基数。分解后的基数越小，分解前后关系模式的空间收益越大，分解后的关系模式越理想。

受信息论的启发，本文认为属性的信息熵、属性间的互信息和KL散度都会反映出属性和属性之间分布的特征，进而评估关系的基数。具体来说在信息论中，属性的信息熵代表了属性取值的不确定性程度，信息熵高的属性可能对应着更加丰富和分散的分布取值，从而可能导致更多的属性基数；属性间的互信息衡量了它们之间的相关性，高互信息的属性对表明属性之间的取值分布更加相似或重叠，即属性间的相关性高，进而影响元组数。KL散度衡量了属性之间的分布差异，较小的KL散度可能意味着属性之间的分布相似度较高，从而影响属性构成的关系的基数。为了更加直观的判断信息论对关系基数的评估，本章设计了如下实验：

本文通过随机生成，模拟了一个具有两个属性列的关系。其属性集，且每个属性均有20个不同的取值。本文统计了模拟出来的关系的占用行数（视为该关系空间占用的量化形式）和两属性间的互信息，并试图分析两者之间可能存在的关系

图4.1展示了随机生成数据5000次得到的互信息值与关系投影基数之间的分布。从图中可以观察到，的基数最小为20（此时和完全函数依赖，I（）=1），最大为400（此时和完全相互独立，I（）=0），尽管受一些数据分布的影响，的基数与它所包含属性间的互信息并不是绝对的函数关系，但图2显示关系实例中属性之间的互信息与基数完全可以近似为强关联，既在属性值域内不同取值个数确定的情况下，属性间的依赖关系（互信息）越强，属性组合的基数越小，关系实例存储占用的空间越小。

基于以上假设，本文将模式分解的目标从最小化 转换为最大化关系模式中各子模式内部属性间依赖关系，即认为模式内部属性的依赖关系越强，模式之间依赖关系越弱，模式存储越节省空间。

图4.1互信息与行数关系图

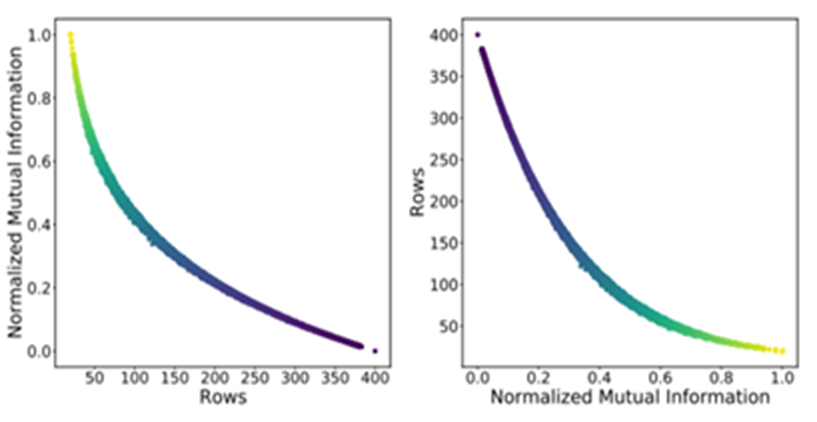
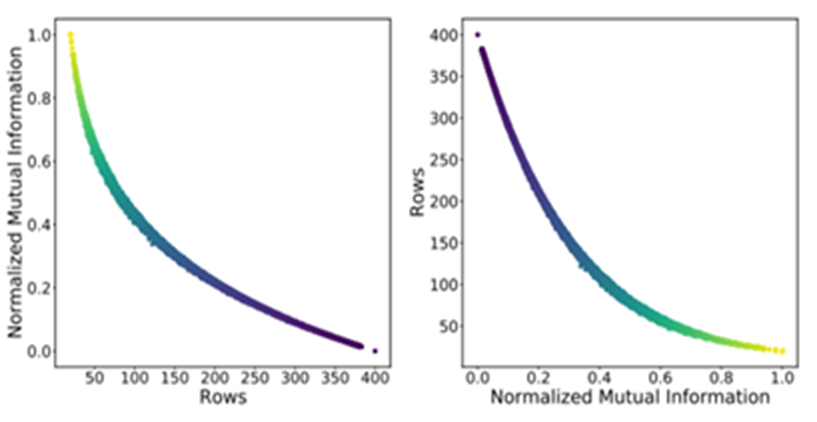


图4.1互信息与行数关系图

图4.1互信息与行数关系图

### 4.2.2归一化互信息设计

基于4.1小节， 本文需要一种度量方式来量化一个模式内部属性之间的依赖关系（已在4.1节中讨论依赖关系与空间增益的关系）。

**定义4.1** （平均互信息）受无权重平均连接方法的启发，本文定义了在关系模式分区集合中的两个分区子集和之间的一种新的多变量互信息：

其中，，是中属性的数量。是子集的第个属性。是式(5.1)中定义的属性和之间的归一化互信息，本文使用平均互信息AMI度量子集的依赖程度。

在实验中观察到，当在数据无缺失值或缺失值很少的情况下，式(4.3)定义的平均互信息可以理想的描述出两个属性集合间的依赖程度。然而，当存在较大比例的缺失值时，会影响到属性列信息熵的计算进而对互信息的结果造成很大的影响。为了解决缺失值对互信息的影响本文扩展了平均互信息的定义来支持缺失值：

**定义4.2** 给定关系和属性集，令表示属性在关系中排除缺失值的投影，熵和互信息表示如下：

其中，是含有缺失值的属性的分布，。

有大量缺失值的关系会浪费大量空间且对查询优化和索引的构建有负面影响(可能缺少主键)。因此应该避免出现大量缺失值的模式。

**定义5.3** 给定属性和，表示和位置关系的惩罚因子的定义如下：

如果和完全重叠（没有缺失值），，如果相互分离（最多缺失值），。本文使用惩罚因子来校正归一化互信息，惩罚会产生大量缺失值的模式：

根据惩罚因子得到修正后的互信息（式4.7）并用它代替式(4.3)中出现的获得。通过最大化可以从中找到具有缺失值的关系上的最优划分，。

## 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

本章从空间收益的视角出发，将模式分解问题转化为一种组合优化问题，旨在最大化数据库的空间利用率。在此过程中，引入了信息论的概念，结合属性之间的信息熵、互信息以及KL散度等指标，对数据库中各属性之间的关联性进行量化评估。通过信息论的视角，本章旨在揭示属性之间的潜在关系，并将这些关系量化为空间收益的度量标准，从而优化数据库的存储效率。

# 第5章 自适应模式分解算法框架

## 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

本章深入研究了所提出的数据驱动的自适应模式分解算法，并通过对比实验以多个维度评估了其性能。具体而言，本文从算法效率、空间利用率、结果准确性以及算法鲁棒性等方面对所提出的算法与其他现有算法进行了全面比较。通过统计相关实验数据并进行深入分析，本文得出结论：在多个关键指标上，所提出的数据驱动的自适应模式分解算法均表现出显著的优势。

## 6.1实验设置

### 6.1.1实验环境

### 6.1.2数据集与对比算法介绍

**数据集：**本文选用 OLTP-Bench 的五个数据集（TPCC，SEATS，SmallBank，Voter，Twitter）及其负载来评估自下而上模式优化算法(AMI-SD)的效果和效率。数据集相关信息如表 1 所示。

**表1 数据集信息**

Table.1 Statistic of datasets

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据集 | 数据描述 | 属性（个） | 元组数 | 数据大小 |
| TPC-C | 订单处理 | 72 | 1,793,390 | 2GB |
| SEATS | 航空票务 | 164 | 17,191,661 | 14GB |
| SmallBank | 银行系统 | 4 | 1,000,000 | 120MB |
| Voter | 选举投票 | 7 | 6,626,540 | 541MB |
| Twitter | 社交网络 | 16 | 1,960,750 | 251MB |

**对比算法** 本文从4类经典的函数依赖挖掘算法中分别选择一种算法来挖掘五个数据集的函数依赖，并将其与BCNF相结合作为模式设计的基准对比算法：

1）Tane[18] (lattice-based)：Tane基于属性值对数据行进行分区以从数据中挖掘函数依赖关系。

2）FastFDs[24] ( difference sets-based )：FastFDs使用启发式驱动的深度优先搜索算法，利用属性列之间的重叠数据挖掘函数依赖关系。

3）HYFD[22]（row and column based hybrid discovery)：HYFD结合传统的行挖掘算法与列挖掘算法的优势，先对采样小部分元组进行列函数依赖挖掘，再通过行挖掘算法对这些函数依赖进行验证。

4）Entropy-based [23]：基于熵的函数依赖挖掘算法将数据转化为二进制向量，并使用熵与互信息度量属性依赖关系，从而实现函数依赖的识别。

Papenbrock T提出的经典的数据驱动模式分解框架Metanomecl1[4]支持函数依赖挖掘算法驱动的模式分解算法，前三种对比算法均在此框架下进行测试；基于熵的发现算法采用作者的源码进行运行测试 [23]。

介绍oltp工作负载、介绍数据集

### 6.1.3评价指标

## 6.2实验结果及其分析

### 6.2.1算法效率对比

### 6.2.2空间收益对比

### 6.2.3模式结果对比

### 6.2.4鲁棒性对比

## 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。