# 支持云租户的关系数据库模式自动优化

## 岳佳琦1, 丁蕊2, 马玲3, 王斌4, 杨晓春5+

东北大学 计算机科学与工程学院， 辽宁省 沈阳市 110169

+ 通信作者E-mail: yangxc@mail.neu.edu.cn, Phn: +86-139-4014-1466

摘 要：关系数据库设计的目标是生成一组关系模式，避免数据存储带来的不必要的冗余。基于函数依赖约束的规范化理论是当前数据库社区中最流行且有效的数据库设计方法，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。因此本文尝试在函数依赖未知的条件下设计数据库模式，提出了一种新的自下而上的模式设计策略。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，以高效的自下而上的层级聚合思想寻找最优模式，从而解决了函数依赖未知条件下模式设计困难的问题。本文提出的算法在通用benchmark数据集上进行了测试，结果表明本文提出的自下而上的数据驱动的模式设计算法能够在函数依赖未知的条件下有效的设计关系模式。

关键词：模式设计；函数依赖；关系型数据库

文献标志码：A 中图分类号：TP399

#### Automatic Optimization of Relational Database Schema for Cloud Tenants

##### Jiaqi Yue1, Rui Ding2, Ling Ma3, Bin Wang4, Xiaochun Yang5+

College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China

**Abstract:** The goal of relational database design is to generate a set of relational schemas that avoid unnecessary redundancy due to data storage. The normalization theory under the constraint of function dependence is the most popular and effective database design method in the current database community. Based on this, the data redundancy and data anomalies caused by functional dependence can be eliminated. Although normalization theory has been well studied by the database community, converting existing data sets into normal forms (BCNF) is still a very complex task, in cloud scenarios where business logic (functional dependencies) is missing and decomposition is a NP-hard problem. Therefore, this paper tries to design database schema under the condition of function dependence unknown, and proposes a new bottom-up schema design strategy. By calculating the dependency relationship between attributes, the algorithm mines the inherent features of data, and finds the optimal pattern through efficient bottom-up hierarchical aggregation, thus solving the difficult problem of pattern design under the unknown condition of function dependence. The algorithm proposed in this paper is tested on general benchmark data set, and the results show that the bottom-up data-driven schema design algorithm proposed in this paper can effectively design relational schemas under the condition of function dependence unknown.

###### **Key words:** Schema Design; functional dependencies; relational databas

**1** 绪论

近年，随着大模型、人工智能、AIoT、云计算等技术的推动，全球数据量正在无限制地扩展和增加，给数据存储方式带来了巨大的挑战[1]。一些数据生产者选择租用云服务器来存储数据，然而一些缺乏数据库规范化理论知识的租户会盲目的将所有数据以单一的大表形式存储在云平台上，这会造成存储空间的浪费。一个良好的关系数据库模式应该具有降低数据冗余、提升查询效率的能力[2]。

关系数据库模式分解任务旨在将一个宽表依据属性之间的函数依赖关系分解为一组无损的更小的关系模式，降低数据冗余的同时提升查询执行效率[3]。但这种传统的关系数据库规范化理论并不适用于云环境下的关系数据规范化，因为函数依赖是未知的。为了在函数依赖未知的关系数据上进行模式分解，Thorsten提出了数据驱动的模式分解[4]，首先利用函数依赖挖掘算法挖掘关系数据中属性列之间暗含的逻辑关系（函数依赖[5]），然后基于这些函数依赖对关系进行分解，设计满足第三范式（3NF[6]）或巴斯范式（BCNF[7]）的模式。但这种数据驱动的模式规范化存在几个问题：（1）函数依赖挖掘算法的可扩展性较差，数据规模（行、列）的增加会导致挖掘算法的效率急剧下降[8]，因此这类数据驱动的模式分解算法效率低下；（2）函数依赖挖掘算法易受脏数据影响而挖掘出错误的函数依赖关系，导致分解出的模式可能不会降低空间开销与冗余；（3）传统的范式分解方法，如3NF、BCNF，是对违反给定范式规则的函数依赖进行分解[9]，直至模式内部没有函数依赖违反指定分范式的规则，这是一个自上而下的局部最优分解策略，不同的分解顺序（函数依赖选择顺序）会导致不同的模式集合，因此传统的范式分解难以为关系R找到全局最优模式。

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支持云租户的关系数据库模式自动优化算法。首先，最优模式被定义为最节省存储空间的模式，然后，基于模式内属性间依赖关系越强模式存储空间越优的假设将空间最优模式与属性之间的依赖关系进行关联。在函数依赖缺失的条件下暴力枚举所有模式并进行评估显然是不可行的，本文尝试设计了一个自下而上的高效的模式设计算法，它可以优先依据属性之间的依赖关系对部分属性进行组合（基于高依赖低存储开销的假设），自下而上的设计也同样规避了传统的自上而下的范式分解算法难以找到最优模式的困境。实验结果表明，本文提出的自下而上的模式优化算法在分解时间和存储空间两个维度均优于经典的数据驱动的范式分解算法（函数依赖挖掘算法与范式分解的结合）。

本文的贡献点主要总结如下：

1）本文首先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）本文设计了一个支持云租户的自下而上的模式优化算法，成功规避了现存的数据驱动的模式分解算法面临的挑战。

3）实验表明，本文提出的基于互信息的自下而上的模式优化算法在时间和空间两个维度上相比现存的数据驱动的模式分解算法都有明显的优势。

4）除了传统模式分解算法带来的空间上的收益之外，实验发现本文提出的自下而上的模式优化算法的额外的优点：算法将依赖程度高的属性逐层聚合，提高了工作负载的查询效率。

**2** 相关工作

**2.1** 范式分解

范式分解（Normalization Decomposition）是关系数据库中广泛应用的模式分解算法，其根据业务逻辑（函数依赖）进行设计。遵循这些函数依赖，可以将数据库模式分解为更小、更规范的子模式，从而消除关系数据中存在的大量冗余信息，提升数据库的查询性能和可维护性[10-13]。

常见的范式包括第一范式(1NF)、第二范式(2NF)、第三范式(3NF)和巴斯范式(BCNF)等。第一范式要求关系模式中的属性必须是不可再分的；第二范式进一步要求模式中非主键属性完全依赖于候选键；第三范式在第二范式基础上要求非主键属性之间不存在传递依赖；巴斯范式在第三范式的基础上要求非主键必须完全依赖于主键属性[14]。

**2.2** 函数依赖挖掘

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，为数据清洗[15,16]、模式分解、查询优化等重要应用场景优化提供依据。

函数依赖挖掘算法可以被分为四大类:

1）属性格(lattice)[17-19]：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)[20,21]：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[22]：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[23]：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。

**3**问题定义

给定一个关系实例R，其属性集合为。模式分解任务尝试将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，，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，则模式S是有效的。

本文将R的最优模式定义为最优化问题：

是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(3.1a)将最优模式定义为导致最大空间增益的模式S。

**4** 基于空间增益的自下而上的模式优化算法

本章对模式分解带来的空间增益进行了量化分析，尝试利用模式内部属性之间依赖程度对最优模式进行建模；提出了一种依赖度量方法（平均互信息AMI）来高效的计算属性之间的依赖关系；基于平均互信息设计了一种自下而上的层级属性聚合算法来探索最优的关系模式。

**4.1** 空间增益建模

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，最大化 等价于：

由式(4.1)可知，影响关系模式空间收益的主要因素是关系模式中各关系在各自属性集上投影的基数。因此本文尝试分析关系投影的基数受哪些因素的影响并进行了以下实验：

本文模拟了一个具有两个属性列的关系，其属性集，且每个属性的distinct值都为20，并使用互信息度量属性间的依赖关系：

图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distinct值确定的情况下，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。

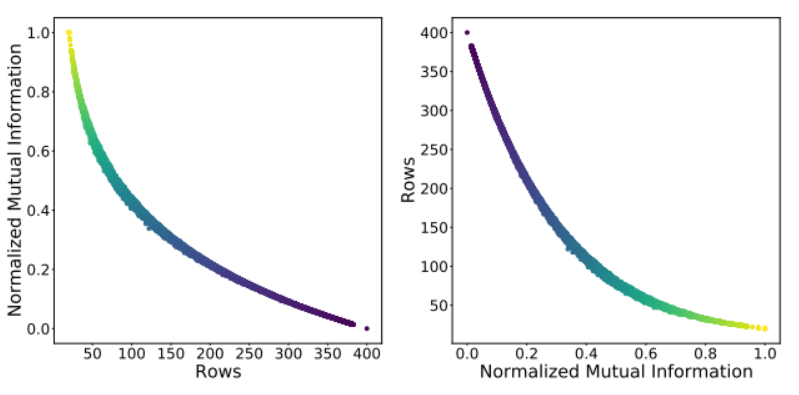


Fig.1 The relation of mutual information and row count

**图1 互信息与行数关系**

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。基于此假设，本文将模式分解过程分为两步：

1）自下而上的基于属性间的依赖程度对其进行聚合，这样能在增强模式内部属性间的关联的同时简化自上而下的贪心策略的复杂度，使两个关系紧密的属性能够更简单的聚合在一起。为了简化计算，本文在聚合操作时不考虑外键约束。

2）设计并添加外键约束，使模式中不相交的关系成为初始关系的空间占用最小的无损分解。

**4.2** 平均互信息

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** 自下而上的属性聚合算法

本节提出了一种基于平均互信息的自下而上的属性聚合算法(AMI-PS)，该算法将关系中每个属性分别看做一个单独的集合，并基于互信息层级向上聚合，直至聚合成为一个大的集合。

在每层聚合过程中，算法会寻找平均互信息最大的两组模式进行合并，逐渐将属性聚合到更粗粒度的关系模式中（不考虑和其他表的外键关系）。在这个过程中，我们将第次聚合过程产生的集合定义为一次划分，其中是属性集合。在聚合过程中产生的所有划分构成了一个最优关系划分候选集。

在层次聚合的过程中平均互信息随聚合过程而递减。在聚合初期，底层的模式分解的过细，引入外键生成无损模式之后会产生过多的空间开销；而在聚合后期，顶层的模式内部属性之间的依赖程度松散，构成的表结构存在大量的空间浪费。因此，本文选择在层次聚合过程中，平均互信息缩减幅度最大的一次划分拥有最优的内部依赖关系

下面给出基于AMI的自下而上的层次聚合算法的伪代码：

**算法1 基于AMI的层次聚合算法**

|  |  |
| --- | --- |
| **算法1：基于AMI的层次聚合算法** | |
| 输入:一个去规范化关系 | |
| 输出: 关系的一个最优划分 | |
| (1) 初始化长度为的数组和,中存储提前计算好的互信息，和保存每轮聚合时的平均互信息和划分。 | |
| (2) | |
| **算法1（续）** | |
| **算法1（续）** | |
|  |  |
|  | |
| (3) | |
| (4) | |
| (5) | |

为了降低该算法时间开销，本文定义了平均互信息更新公式（4.8），使用该公式允许在算法中只算一次属性间的互信息，大大提高了算法的效率：

**定义4.2** （平均互信息更新公式）当算法将分区和合并在一起，新形成的分区与任何其他分区之间的AMI是,和,的加权平均互信息：

其中，，。

**4.4** 最优模式生成

为了将4.3中得到的空间最优划分转换为最优的无损关系模式还需要添加外键约束使中的关系能够无损连接成为初始关系。本文从空间最优的角度考虑，选择内每个属性集合中能够唯一标识一条记录的最小属性集作为主键，此时有且。

受Kruskal算法的启发，本文将各关系的初始主键视为图中的单个节点，并通过构建一个连通图实现了无损关系设计。与Kruskal算法不同的是，在选择连接两个节点（表示两个初始主键 和）时，本文将具有较少基数的节点（）添加到另一个节点所属的关系中，得到新的关系 并更新 的初始主键。

**定理4.2** （AMI-PS无损设计）给定最优分区 具有值集 和每个子分区的初始主键 ，将每个，视为图中的单个节点，图中两个节点和之间的节点距离定义如下：

其中，是估计出的元组数，是连接前键节点的实际的单元格数。两个连接的键节点之间的距离应该尽可能小，这表示冗余的数据较少。

随机选择距离最小的前个节点对进行连接，距离越小，被选择的机会越大。该算法将迭代执行次。通过AMI-PS无损设计，的子集能无损连接在一起，同时确保子模式之间的连接紧凑，不浪费过多的单元格。本文后续称AMI-PS与无损设计为AMI-SD算法。

**4.5** 复杂度分析

基于平均互信息的分层模式分解策略的复杂度为，其中n是属性的数量。算法2的运行时间主要取决于熵和互信息的计算，因此本文提出了一些规则来尽可能提高计算效率，结果在实验中说明。

**5** 实验结果与分析

本节从分解算法的效率（分解时间）、分解算法的效果（存储空间）与分解算法的其他收益（对负载的影响）三个方面进行了实验测试与分析。

**5.1** 实验设置

**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]。

**运行环境**：本文提出的AMI-SD方法和四种对比算法均在64GB内存的Linux服务器（Ubuntu 20.04.3 LTS）上进行实验测试。

**5.2** 模式分解算法性能分析

**5.2.1** 分解算法的效率

表2测试了基于FDDs的BCNF范式分解算法在五个数据集上挖掘函数依赖个数、分解后关系数和执行时间（由于算法复杂度过高，且数据集行列数基数过大，因此对于某些算法采用抽样的策略进行实验，并对运行时间超过24h的判定为无法在有效时间内完成分解的情况），并与本文提出的AMI-SD算法进行了对比分析。从表2的实验结果可以得到以下结论：（1）在五个数据集上，本文提出的AMI-SD算法的分解时间均显著优于其他函数依赖挖掘算法的运行时间，函数依赖挖掘算法在行列较多的情况下可扩展性明显较差，运行缓慢；（2）函数依赖挖掘算法不仅可扩展性较差，还有分解无效的可能，如在seats和TPCC两个属性较多的关系实例上进行函数依赖挖掘耗费了大量的时间但却分解分解不出合理的模式；（3）本文提出的AMI-SD分解算法可扩展性较好，高效的支持不同数据规模下的模式分解

**表2 分解算法效率**

Table.2 Efficiency of schema decomposition methods

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 评测指标 | HYFD | Tane | FastFDs | Entropy | AMI-SD |
| Voter | # FDs | 6 | 6 | 7 | 23 | - |
| # Tables | 4 | 4 | 5 | 5 | 3 |
| Running time (ms) | 89,231 | 174,032 | 162,951 | 186,425[25k] | 60,273 |
| SmallBank | # FDs | 2 | 2 | 3 | 11 | - |
| # Tables | 1 | 1 | 1 | 3 | 1 |
| Running time (ms) | 12,769 | 11,131 | 42,275 | 4,369[25k] | 1,705 |

**表2（续）**

Table.2 (continued)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 评测指标 | HYFD | Tane | FastFDs | Entropy | AMI-SD |
| Twitter | # FDs | 5 | 5 | 15 | \* | \_ |
| # Tables | 2 | 2 | 2 | \* | 3 |
| Running time (ms) | 40,257 | 46,341 | 1,574,157[25k] | \* | 6,125 |
| Seats | # FDs | 7 | \* | 2206478 | \* | - |
| # Tables | 1 | \* | 1 | \* | 7 |
| Running time (ms) | 6,519,025 | \* | 1,869,585[1k] | \* | 205,999 |
| TPC-C | # FDs | 253,423 | \* | 72,312 | \* | - |
| # Tables | 1 | \* | 1 | \* | 7 |
| Running time (ms) | 39,493,399 | \* | 394,933,990[25k] | \* | 2,629,806 |

**注：**其中# FDs：挖掘出的函数依赖数量；# Tables：分解得到的模式数量，“\*”：对应方法可扩展性较差，运行超过24h没有得到分解结果，，“-”：AMI-SD不需要函数依赖所以无此项信息，[n]：扩展性较差，只能在n行采样数据上挖掘FD得到部分函数依赖。

**5.2.2** 分解算法的效果

表3统计了AMI-SD分解算法得到的模式与其他四种对比算法分解得到的模式在空间上的表现，空间增益的计算方式为分解后节省的存储空间与原始关系存储空间大小的比值。从表3的结果可以得到：在五个数据集上，AMI-SD分解得到的空间增益与其他四种对比算法的空间增益相比是持平或更优的。持平的主要原因是该关系实例含有的属性较少，如SmallBank仅有4列属性，分解并不能带来更好的空间增益反而会增加外键的空间开销，因此会维持原有表格的结构。对于属性较多的关系实例，如seats和TPCC，AMI-SD的分解效果要明显优于对比算法，在seats上存储空间节省了60%，而对比算法则因函数依赖误判、空值过多等原因并没有实现合理的分解。综上所述，AMI-SD分解算法在各种关系实例上都能取得较好的空间增益。

**5.2.3** 负载效率评测

除对AMI-SD分解算法的效率与效果进行测试外，ANI-SD分解算法对查询负载的影响也是值得关注的。表4统计了不同分解算法得到的模式对查询负载的影响（查询语句顺序扫描的平均运行时间），从实验结果可以看出，AMI-SD分解算法得到的模式在降低存储空间的同时也对下游查询任务的效率进行了优化，这主要是因为AMI-SD降低了存储代价，进而例降低了IO代价。

**表3 空间增益对比**

Table.3 Effectiveness of schema decomposition methods

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 空间增益指标 | HYFD | Tane | FastFDs | Entropy | AMI-SD |
| Voter | 空间增益率 | 39.1% | 39.1% | 5.9% | 16% | 20.3% |
| SmallBank | 空间增益率 | 0% | 0% | 0% | -56% | 0% |
| Twitter | 空间增益率 | 0% | 0% | 19.9% | 0% | 24.43% |
| Seats | 空间增益率 | 0% | 0% | 0% | - | 60.45% |
| TPC-C | 空间增益率 | 0% | 0% | 0% | 0% | 30.46% |

**表4 负载的平均运行时间（顺序扫描，ms）**

Table.4 Average running time of workload (seqscan, ms)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 查询语句（条） | HYFD | Tane | FastFDs | Entropy | AMI-SD | |
| Voter | 500 | 15.2360 | 15.2360 | 15.2360 | 325.8472 | | 0.1470 |
| 1,000 | 15.1355 | 15.1355 | 15.1355 | 327.5623 | | 0.1052 |
| 5,000 | 15.2155 | 15.2155 | 15.2155 | 326.8179 | | 0.1041 |
| SmallBank | 5,00 | 66.7298 | 67.7688 | 69.7248 | 102.5146 | | 66.7298 |
| 1,000 | 66.0990 | 70.9375 | 66.0990 | 102.1217 | | 66.0990 |
| 5,000 | 66.9530 | 74.3267 | 66.9530 | 100.0236 | | 66.9530 |
| Twitter | 500 | 182.9434 | 182.9434 | 100.7041 | 182.9434 | | 8.0924 |
| 1,000 | 179.8658 | 179.8658 | 100.7601 | 179.8658 | | 10.3486 |
| 5,000 | 172.6595 | 172.6595 | 103.6912 | 172.1595 | | 11.5892 |
| Seats | 500 | 555,917.1830 | 555,917.1830 | 555,917.1830 | 555,917.1830 | | 2,501.9773 |
| 1,000 | 556,505.8457 | 556,505.8457 | 556,505.8457 | 556,505.8457 | | 3,744.9724 |
| 5,000 | 560,351.7274 | 560,351.7274 | 560,351.7274 | 560,351.7274 | | 3,694.4451 |
| TPC-C | 500 | 9,144.5618 | 9,144.5618 | 9,144.5618 | 9,144.5618 | | 877.9305 |
| 1,000 | 4,442.8057 | 4,442.8057 | 4,442.8057 | 4,442.8057 | | 856,4562 |
| 5,000 | 4,050.3307 | 4,050.3307 | 4,050.3307 | 4,050.3307 | | 895.2870 |

**5** 结束语

本文提出了一种基于互信息(AMI)的自下而上的数据库模式分解算法，并在五个数据集上与现有的四种基于函数依赖挖掘的模式分解算法进行了对比实验，均得到了更优的实验结果。实验过程中发现算法对脏数据的鲁棒性还需要进一步提升，这是未来继续研究的方向。

**参考文献：**

[1]GHOSH I. AIoT: when artificial intelligence meets the Internet of things[J]. Visual Capitalist, 2020, 12.

[2]WIEDERHOLD G. Mediators in the architecture of future information systems[J]. Computer, 1992, 25(3): 38-49.

[3]CODD E F. Further normalization of the data base relational model[J]. Data base systems, 1972, 6: 33-64.

[4]PAPENBROCK T, NAUMANN F. Data-driven Schema Normalization[C]//EDBT. 2017, 17: 342-353.

[5]HUHTALA Y, KARKKAINEN J, PORKKA P, et al. Efficient discovery of functional and approximate dependencies using partitions[C].Proceedings 14th International Conference on Data Engineering. IEEE, 1998: 392-401.

[6]DELOBEL C. Normalization and hierarchical dependencies in the relational data model[J]. ACM Transactions on Database Systems (TODS), 1978, 3(3): 201-222.

[7]BOYCE, R. F., CODD, E. F. A normal form for database relations based on attribute dependencies. Communications of the ACM, 1974, 17(7), 387-395.

[8]Fagin R. A normal form for relational databases that is based on domains and keys[J]. ACM Transactions on Database Systems (TODS), 1981, 6(3): 387-415.

[9]MANNILA H, TOIVONEN H, INKERI VERKAMO A. Discovery of frequent episodes in event sequences[J]. Data mining and knowledge discovery, 1997, 1: 259-289.

[10] KENIG B, MUNDRA P, PRASAAD G, et al. Mining ap-proximate acyclic schemes from relations[C]. Proceedings of the 2020 ACM SIGMOD International Conference on Man-agement of Data. 2020: 297-312.

[11] KÖHLER H, LINK S. SQL schema design: foundations, normal forms, and normalization[C]. Proceedings of the 2016 International Conference on Management of Data. 2016: 267-279.

[12] LINK S, WEI Z. Logical schema design that quantifies update inefficiency and join efficiency[C].Proceedings of the 2021 International Conference on Management of Data. 2021: 1169-1181.

[13] WEI Z, LINK S. Embedded functional dependencies and data-completeness tailored database design[J]. ACM Transac-tions on Database Systems (TODS), 2021, 46(2): 1-46.

[14] ARMSTRONG W W. Dependency structures of data base relationships[C].IFIP congress. 1974, 74: 580-583.

[15] FAN W, BOHANNON P, GEERTS F, et al. Conditional functional dependencies for data cleaning[C].Data Engineering, 2007, IEEE 23rd International Conference on. IEEE, 2007 (746-755).

[16] REKATSINAS T , CHU X , ILYAS I F , et al. HoloClean: Holistic Data Repairs with Probabilistic Inference[J]. Proceed-ings of the VLDB Endowment, 2017, 10(11).

|  |  |
| --- | --- |
|  | 岳佳琦（2000—），男，河北保定人，硕士研究生，主要研究方向为模式分解、基数估计等。  YUE Jiaqi, born in 2000 year, M.S. candidate, His research interests include schema decomposition, cardinality estimate, etc. |
|  | 丁蕊（1995—），女，辽宁朝阳人，博士研究生，主要研究方向为模式分解、推荐系统等。  Ding Rui, born in 1995 year, Ph.D. candidate, Her research interests include schema decomposition , recommender systems , etc. |
|  | 马玲（1999—），女，河南舞阳人，硕士研究生，主要研究方向为聚类算法、图论等。  NAME Name, born in 1999 year, M.S. candidate, Her research interests include clustering algorithm , graph theory , etc. |
|  | 王斌（1972—），男，辽宁沈阳人，博士，教授，主要研究方向为数据管理、数据质量等。  WANG Bin, born in 1972 year, Ph.D., professor, His research interests include data management, data quality, etc. |
|  | 杨晓春（1973—），女，辽宁沈阳人，博士，教授，主要研究方向为数据质量管理、数据隐私保护等。  YANG Xiaochun, born in 1973 year, Ph.D., professor, Her research interests include data quality management, data privacy preservation, etc. |

[17] ABEDJAN Z, SCHULZE P, NAUMANN F. DFD: Efficient

functional dependency discovery[C]. Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. 2014: 949-958.

[18] HUHTALA Y, KÄRKKÄINEN J, PORKKA P, et al. TANE: An efficient algorithm for discovering functional and ap-proximate dependencies[J]. The computer journal, 1999, 42(2): 100-111.

[19] ILYAS I F, MARKL V, HAAS P, et al. CORDS: Automatic discovery of correlations and soft functional dependencies[C]. Proceedings of the 2004 ACM SIGMOD international confer-ence on Management of data. 2004: 647-658.

[20] NOVELLI N, CICCHETTI R. Fun: An efficient algorithm for mining functional and embedded dependencies[C]. Data-base Theory—ICDT 2001: 8th International Conference Lon-don, UK, January 4–6, 2001 Proceedings 8. Springer Berlin Heidelberg, 2001: 189-203.

[21] LOPES S, PETIT J M, LAKHAL L. Efficient discovery of functional dependencies and armstrong relations[C]. Advances in Database Technology—EDBT 2000: 7th International Con-ference on Extending Database Technology Konstanz, Ger-many, March 27–31, 2000 Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000: 350-364.

[22] PAPENBROCK T, NAUMANN F. A hybrid approach to functional dependency discovery[C]. Proceedings of the 2016 International Conference on Management of Data. 2016: 821-833.

[23] MANDROS P, KALTENPOTH D, BOLEY M, et al. Discovering functional dependencies from mixed-type data[C]. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020: 1404-1414.

[24] WYSS C, GIANNELLA C, ROBERTSON E. FastFDs: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract[C]. Data Warehousing and Knowledge Discovery: Third International Conference, DaWaK 2001 Munich, Germany, September 5–7, 2001 Proceedings 3. Springer Berlin Heidelberg, 2001: 101-110.