数据立方体与频繁项集的统一计算框架研究

徐静文[1)] 游进国[1),2)] 王全鹞[1)] 黄星瑞[1)] 贾连印[1),2)]

[1)](昆明理工大学信息工程与自动化学院 昆明 650500)

[2)](云南省人工智能重点实验室 昆明 650500)

摘 要 数据立方体和频繁项集挖掘分别是数据仓库和数据挖掘领域的重要技术,已开展了大量的相关研究工作,取得了较好的进展.数据立方体和频繁项集挖掘依据各自的数据单元和项集构造了类似的代数格(Lattice)结构;数据立方体的等价类上界单元与频繁项集挖掘的闭项集也是相对应的.如果能够论证二者的统一性,则可以为彼此提供更广泛的研究思路,有利于两种技术的相互促进,如:在数据库中利用冰山立方体计算实现频繁项集挖掘来避免数据迁移、利用频繁项集挖掘算法优化数据立方体计算等.之前的工作没有将二者系统地结合起来研究,也没有建立二者之间较为完整的联系.本文在深入研究数据立方体的计算和频繁项集挖掘的过程后,将二者有效地结合在一起,提出了统一的计算框架,给出了二者众多计算性质和方法之间的映射关系,进行了相关概念泛化,具体地建立了冰山立方体、浓缩立方体和商立方体等主要数据立方体计算与相应频繁项集挖掘方法的对应关系.通过算法和实验进一步论证统一计算的有效性:(1)将频繁项集挖掘事务集导入关系数据库,用冰山立方体计算方式进行频繁项集挖掘,从而在数据库中用标准的或扩展的SQL可以实现对关系表进行频繁项集挖掘;(2)验证了浓缩立方体与频繁项集挖掘的统一性并对比了计算效率;(3)将基本表转换为频繁项集挖掘事务集,引入高效的频繁项集挖掘算法LCM计算商立方体,以提升数据立方体计算效率.在公开的真实数据集和人工合成的数据集上验证二者结合、统一计算的正确性,通过改变元组数、维数和倾斜度进行对比验证有效性.实验发现,在大数据集上可令时间效率提升高达92%.

关键词 数据立方体;频繁项集挖掘;格结构;统一计算方法;计算效率

中图法分类号 TP18 **DOI号** 10.11897/SP.J.1016.2023.00780

Unified Computing Framework for Data Cubes and Frequent Itemsets

XU Jing-Wen[1)] YOU Jin-Guo[1),2)] WANG Quan-Kun[1)] HUANG Xing-Rui[1)] JIA Lian-Yin[1),2)]

[1)](*Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming* 650500)

[2)](*Yunnan Key Laboratory of Artificial Intelligence, Kunming* 650500)

Abstract Data cube and frequent itemset mining are essential technologies in the field of data warehouse and data mining respectively. A lot of relevant research work has been conducted, and impressive progress has been achieved. Data cube and frequent itemset mining construct similar algebraic lattice structures according to their data cell and itemset. Simultaneously, the upper bounds of the equivalent class of the data cube correspond to the closed itemset of frequent itemset mining. If the unity of the two lattice structures can be argued, they can provide broader research ideas for each other and facilitate the mutual promotion of the two techniques. For example, we can use iceberg cube computing to implement frequent itemset mining to avoid data migration in databases, and use frequent itemset mining algorithms to optimize data cube computing. Previous

收稿日期:2022-05-03;在线发布日期:2022-11-05.本课题得到国家自然科学基金项目(No. 62062046, No. 61462050)资助.**徐静文**,硕士研究生,主要研究领域为数据挖掘、大数据.E-mail:inweun@stu.kust.edu.cn. **游进国**(通信作者),博士,教授,中国计算机学会(CCF)高级会员,主要研究领域为大数据、云计算、数据仓库.E-mail:jgyou@126.com. **王全鹞**,硕士研究生,主要研究领域为大数据、云计算. **黄星瑞**,硕士研究生,主要研究领域为数据挖掘、机器学习. **贾连印**,博士,副教授,中国计算机学会(CCF)会员,主要研究领域为数据库、数据挖掘、信息检索、并行计算.

studies have not studied the two concepts with systematic combination, nor have they established a complete connection between them. After intensely studying the computation of data cube and the process of frequent itemset mining, this paper combines data cube and frequent itemset mining effectively. The paper proposes a unified computation framework, and gives the mapping relationship between multitudes of computational properties and methods of two lattice structure. Therefore, the high-performance lattice structure computation methods and application algorithms in the two fields can be integrated, thus improving the performance of lattice structure usage and enhancing the efficiency and accuracy of data analysis. On the basis of these results, related concept generalization was performed. Specifically, the corresponding relationship between the computation method of classic data cubee such as the iceberg cube, the condensed cube, and the quotient cube and the corresponding frequent itemset mining method is established. The effectiveness of the unified computation framework is further demonstrated by algorithms and experiments. First, the transaction datasets of frequent itemset mining are imported into the relational database, and the frequent itemset mining is performed with the iceberg cube computation method, so that frequent itemset mining of relational tables can be implemented in the database with standard SQL-92 or extended SQL. Secondly, the unification of the condensed cube and the frequent itemset mining is verified by experiments and the computation efficiency is compared. Finally, the base table is converted into a transaction dataset of frequent itemset mining, and LCM, an effective algorithm for frequent itemset mining, is introduced to implement the quotient cube computation to improve the computation efficiency of the data cube. At the meanwhile, we give an example to illustrate and explain the unified algorithm. The correctness of the combination and unified framework is verified by the experiments both on the publicly available real datasets and the synthetic datasets. Besides, the effectiveness is compared and verified by changing the number of tuples, dimensions, and skewness. It is found that the time efficiency can be improved by up to 92% on large datasets.

Keywords data cube; frequent itemset mining; lattice structure; unified computation method; computation efficiency

1 引 言

数据立方体[1-2]是数据仓库和联机分析处理(Online Analytical Processing, OLAP)研究领域一种重要的数据模型. 为提高数据仓库中联机分析和决策支持查询的性能, Jim Gray等学者[1]在1996年首次提出了CUBE BY算子, 并给出数据立方体的相关定义. 数据立方体泛化了关系操作符GROUP BY, 对每一种可能GROUP BY的属性组合都进行了聚集计算.

面向大规模数据时, 如何对数据立方体进行高效计算和压缩, 是多年来数据仓库乃至大数据分析研究领域的一个难题. 因此, 利用数据立方体内部结构的性质来降低立方体计算量的研究仍然有必要. 经典的数据立方体计算方法有基于Apriori剪枝算法的冰山立方体[3](Iceberg Cube)、浓缩立方体[4](Condensed Cube)、侏儒立方体[5](Dwarf Cube)、共享元组方式的商立方体[6](Quotient Cube)以及壳立方体[7](Shell Cube).

冰山立方体用于回答冰山查询问题, 能够按照用户的需求计算出满足条件的方体. 冰山查询由Fang等[3]首次提出, 是指在每个基本GROUP BY操作中添加同一个HAVING限制条件后计算, 例如: SELECT S, P, SUM(*sales*) FROM *basetable* GROUP BY S, P HAVING COUNT(*)>=10为冰山查询. Beyer等[8]提出了相应的冰山立方体, 并首次采用自底向上(Bottom-Up Construction, BUC)方法进行计算. BUC方法从0维方体开始计算, 若某数据单元不满足HAVING条件(即最小支持度阈值), 根据反单调性, 以该单元为根的整个分枝也将违反该条件, 因此可以对该分枝进行剪枝, 从而减少

数据立方体计算的计算量. 冰山立方体不仅能够有效减小数据立方体的体积,还能够忽略数据立方体中存在意义不大的方体,仅保留携带信息较多的方体.

商立方体[6]的核心思想是将覆盖相同元组集的数据单元归为同一个等价类,该等价类中的所有单元可以压缩为等价类的上界单元来保存,其余数据单元可以通过等价关系推导得到. Lakshmanan等[9]进一步提出了商立方体的树形简洁结构QC-Tree,能够压缩和索引商立方体,并给出了商立方体的查询和增量式维护的算法. 而浓缩立方体[4]、侏儒立方体[5]通过寻找数据单元间相同的前缀或后缀来进行压缩,均为商立方体的变体. 由于商立方体能够在压缩数据立方体尺寸的同时,不丢失数据立方体上的语义信息,因此商立方体是较为理想且重要的数据立方体约简方法.

关联规则挖掘[10]是数据挖掘领域重要的研究方法之一,能够从事务数据库中发现事务间有价值的关联关系. 频繁项集/频繁闭项集/频繁极大项集挖掘作为关联规则挖掘中的重要一步,该部分的算法性能很大程度上影响了关联规则挖掘的性能. 由于频繁项集挖掘过程满足反单调性,因此可以通过剪枝来提高频繁项集生成的效率. 同时,项集之间大多具有包含关系,因此大量频繁项集挖掘算法通过定义双亲-孩子关系来生成树形结构,从而缩短算法的运行时间[11-16].

频繁项集挖掘的过程所具有的层次结构与数据立方体的计算方法具有相同的代数格结构. 本文根据频繁项集挖掘和数据立方体所呈现出的性质,将二者联系起来,发现并研究其中的关联关系. 例如:频繁项集挖掘过程中对项集的剪枝与冰山立方体中对方体的剪枝均依据用户给定的最小支持度阈值;频繁项集挖掘中项集的支持度对应于数据立方体单元的COUNT值;商立方体中等价类的上界单元与频繁闭项集挖掘的闭项集具有相似的性质.

以表1为例,其展示了一个具有3个维度属性(S, P, T)和1个度量属性*sales*的基本表R,该基本表有3个基本元组. 基本表R通过一定方式向事务集映射后得到的形式如表2.

令最小支持度阈值$min_sup=2$,取冰山查询度量为COUNT,在基本表R上计算冰山立方体;在R的事务集形式R'上计算满足最小支持度阈值的频繁项集. 计算结果如图1所示,图1(a)为满足支持度阈值的冰山立方体(节点表示满足支持度阈值的数据单元);图1(b)为满足支持度阈值的频繁项集(节点为二元组,支持该项集的事务*tid*集以及满足支持度阈值的项集). 可见在相同的数据集和最小支持度阈值的情况下,冰山立方体与频繁项集具有相同的计算结果.

表1 基本表R

S	P	T	*sales*
$S1$	$P1$	$T1$	3
$S1$	$P2$	$T1$	6
$S2$	$P1$	$T2$	9

表2 基本表R的事务集形式R'

tid	S	P	T
1	A	B	C
2	A	D	C
3	E	B	F

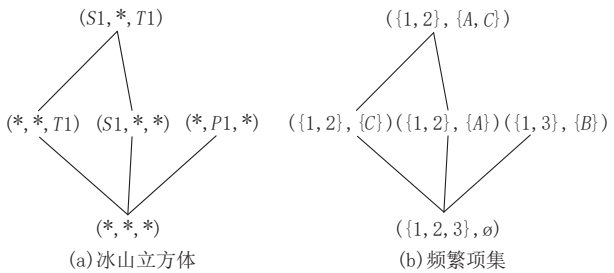


图1 $min_sup=2$时计算所得冰山立方体与频繁项集

如果能够证明数据立方体与频繁项集计算的统一性,则可以将二者在组合计算上的优势进行结合,利用一方的高效算法或有效性质来促进对另一方的研究. 例如,论证数据立方体与频繁项集格的一致性后,可以将数据立方体的其余聚集函数应用于频繁项集格的计算中,向用户提供不同的度量聚集值来衡量项集的"频繁"与否;论证商立方体与闭项集格的统一性,可以将闭项集计算的加速特性应用至商立方体计算中,从而提升商立方体算法的性能.

之前的方法没有将数据立方体计算和频繁项集挖掘系统地结合起来研究,建立二者之间较为完整的关系,进行较为深入的融合. 本文通过分析数据立方体和频繁项集挖掘的性质和计算过程,从代数格的角度出发,将二者相结合并进行统一研究,给出二者统一性的论证,为后续二者的融合算法提供理论支撑. 主要工作及贡献如下:

（1）提出数据立方体基本表与频繁项集挖掘事务集的映射规则．频繁项集格中的项集均能在数据立方体里找到特定的元组或数据单元与之相对应，并进一步进行了相关概念泛化；

（2）首次系统地建立了冰山立方体、浓缩立方体、商立方体等主要数据立方体计算和频繁项集挖掘之间的对应关系，论证了它们本质上是等价的，从而建立二者之间一系列算法的关系，为二者算法融合提供依据；

（3）给出二者融合计算的示例．将频繁项集计算映射为冰山立方体计算，使得关系数据库中能够用标准的SQL-92或扩展的SQL进行频繁项集挖掘；利用冰山查询的聚集函数泛化频繁项集挖掘中支持度的计算，验证泛化频繁项集与冰山立方体计算的正确性；引入高效的项集挖掘算法LCM[13-15]实现商立方体计算，以验证二者统一计算的可行性；

（4）在真实数据集和合成数据集上进行实验，实验结果表明冰山立方体计算与频繁项集挖掘具有相同的计算结果，证明了映射的有效性并对比了两种算法的运行时间；LCM算法通过重用闭算子的计算结果和生成子表，减小了检索闭项集（商立方体等价类上界单元）的探索空间，使得LCM算法在计算商立方体时的性能相对于经典的商立方体计算方法有较大的提升，在大型数据集上时间效率最高可提升92%．

2 相关工作

数据立方体能够提供数据的多维视角，并且允许用户预计算和快速获取汇总数据，成为提升大数据分析性能的重要技术之一．

目前，数据立方体计算方法的相关研究主要是在经典计算方法[3-7]上的改进．文献[17]提出了封闭立方体，核心思想是只保留商立方体的等价类上界从而降低存储开销．文献[18]在商立方体的基础上提出了下钻立方体，从语义的角度考虑立方体存储．文献[19]将形式概念分析引入数据立方体的研究，提出了概念格与商立方体的等价性，并根据该性质对商立方体进一步约简，提高商立方体的计算效率．文献[20]根据浓缩立方体和商立方体的核心思想，提出了基于函数依赖的数据立方体约简方法，能够在保留语义的同时对冗余方体进行剪枝．文献[21]在商立方体的基础上，根据查询能力和方体尺寸对等价类进行剪枝，从而减少空间开销并提升查询效率．上述数据立方体约简方法都是通过对商立方体等经典计算方法进行剪枝来降低时间和空间开销，而未进一步考虑数据之间内在的层次关系，根据数据的关联关系来加快数据立方体计算．

此外，由于现阶段硬件存储和分布式计算的能力得到了较大的提升，部分学者利用硬件和分布式计算来提高数据立方体的计算速度[22-24]．还有部分学者关注更多应用场景下的数据立方体计算，例如Probabilistic Cube[25]、Percentage Cube[26]等．

关联规则挖掘寻找同一时间中出现的不同项的相关性[27]，例如分析用户的“购物篮”中不同商品之间的关联关系．Agrawal最早提出了频繁项集挖掘的Apriori算法[28]，该算法以逐层搜索的迭代方法为基础，收集满足最小支持度的项集，但缺点是会生成许多冗余的候选项，并且需要多次扫描数据库．而FP-Growth[11]算法虽然不需要生成候选集，能够将整个数据库存储在一个高度浓缩的数据结构中，但是该算法构建和使用FP树的过程较为复杂．LCM（Linear Time Closed Item Set Miner，线性时间闭项集挖掘）[13-15]是一种高效枚举频繁闭项集的算法，相对于之前的算法，通过多种优化方法的组合，LCM的算法性能有较大的提升，被认为是这项任务最有效的算法之一．文献[15]从形式概念分析的角度对LCM算法进行了完整的描述，给出了LCM关于数据预处理、构造FP树、剪枝的相关算法．

数据立方体作为多维数据和联机分析处理的重要技术，被广泛应用于数据分析领域．例如：数据立方体在大数据OLAP开源项目Kylin和Druid中发挥着重要作用；同样，频繁项集挖掘的算法在经过不断发展和改进后，大大提升了计算效率[12,29]，并具有了更广泛的应用场景，如空间并置模式挖掘[30]、频繁模式挖掘的查分隐私保护[31-32]、异常子群发现技术[33]、图[34-36]或树[37]等非结构化数据的频繁模式挖掘．尽管二者均为各自领域的重要技术，但目前仅有少量工作将频繁项集挖掘和数据立方体结合起来进行研究．文献[38]提出直接从数据立方体中挖掘多维数据间关联规则的高效算法．文献[39]在OLAM体系结构的基础之上，提出了一种基于数据立方体的多维多层挖掘的系统结构，总结分析了现有的基于数据立方体的多维多层关联规则的挖掘算法，并进行了一定的改进．文献[40]通过分析特定时期的用户查询，找到频繁的谓词项集，根据最小支

持度阈值和最大立方体尺寸对数据立方体进行分区.

上述工作具有一定的局限性:均是借助数据立方体来辅助频繁项集挖掘算法或引入频繁项集挖掘发现数据立方体的多维数据间关联,没有分析二者的内在关联,也没有将二者进行全面的映射以实现统一研究.数据立方体计算与频繁项集计算二者各自提出了大量的优化算法,然而当前二者领域算法没有系统地进行横向关联比较,孰优孰劣未有定论,本文通过研究频繁项集挖掘过程和数据立方体计算方法的内在联系,不仅论证了二者较多主要算法在本质上是一致的,而且还以各自领域较优的商立方体算法和LCM算法进行了比较,发现用关联挖掘LCM算法优化的数据立方体计算相较商立方体能够更大地提升数据立方体计算效率.

3 基本定义

3.1 数据立方体

给定基本表$R=(D_1,D_2,\cdots,D_n,M)$,其中D_i为维度属性,M为度量属性集合.维度属性的一个组合构成了一个方体,表示对该属性组合进行GROUP BY聚集计算.相应地,维度属性值上的一个组合构成一个数据单元(简称数据单元):$(d_1,d_2,\cdots,d_n)$,$(d_1,d_2,\cdots,d_{n-1},ALL)$,$\cdots$,$(ALL,\cdots,ALL)$.其中$ALL$为维度属性的特殊取值,表示在该维上进行了聚集,后文中简记为*.数据单元间的偏序关系定义为:设$c_1(x_1,x_2,\cdots,x_n)$和$c_2(y_1,y_2,\cdots,y_n)$为单元集中的两个单元,对于所有i,如果每当$x_i\neq ALL$时,都有$y_i=x_i$成立,则有$(x_1,x_2,\cdots,x_n)\preccurlyeq(y_1,y_2,\cdots,y_n)$.此时以立方体单元作为节点的哈斯图(Hasse Diagram)中,存在一条从c_1到c_2的向上的路径.在数据立方体语义上,c_1沿该路径下钻至c_2(或c_2可以上卷至c_1).若不存在c_3,使得$c_1\preccurlyeq c_3\preccurlyeq c_2$,则称单元$c_1$为单元$c_2$的孩子,$c_2$为$c_1$的双亲.

定义1. 数据立方体. 数据单元的集合L及单元间的偏序关系$\preccurlyeq$构成了数据立方体$<L,\preccurlyeq>$.

定义2. 基本单元组[4](Base Single Tuple,BST).给定一个维度属性集$SD\subset\{D_1,D_2,\cdots,D_n\}$,当基本表在$SD$上对元组进行划分(Partition)时,$r$为一个划分中的唯一元组,则称$r$是$SD$上的一个基本单元组,而$SD$称为$r$的单维.

例1. 以表1为例,设$SD=\{S,T\}$,基本表在SD上进行划分时,被划分为两组:$\{(S1,P1,T1),(S1,P2,T1)\}$和$\{(S2,P1,T2)\}$,元组$r=(S2,P1,T2)$为第二组中的唯一元组,故$r$为$SD$上的一个基本单元组.此时元组$r$在$SD$上浓缩了单元$\{(S2,P1,T2),(S2,*,T2),(S2,*,*),(*,*,T2)\}$.

定义3. 浓缩立方体[4]. 一个浓缩立方体$<L_c,\preccurlyeq>$中所有的单元$c\in L_c$满足以下条件:

(1)所有非基本方体单元$c_1(x_1,x_2,\cdots,x_n,m_1)$均不存在另一单元$c_2(y_1,y_2,\cdots,y_n,m_2)\in L_c$,使得$m_1=m_2$,并且对于$\forall D_i\in SD$有$x_i=y_i$,$\forall D_j\notin SD$有$y_j=ALL$;

(2)浓缩立方体中的所有单元和它们所浓缩的单元的并集能够构成完整的数据立方体$<L,\preccurlyeq>$.

浓缩立方体依据基本单元组的概念对数据单元进行划分,上述条件(1)保证了立方体基于基本单元组理论进行浓缩.显然,根据基本单元组进行划分后,对于任意聚集函数,每个划分中的单元都具有相同的聚集值,因为它们都由同一基本单元组聚集得到.

商立方体(Quotient Cube,QC)依据覆盖等价关系将单元划分为不同的等价类,对于用户给定的任意聚集函数,每个类中的单元具有相同的聚集值,用等价类的上界来概括类中的所有单元.

定义4. 基本元组集[21](Base Tuple Set,BTS).若存在一条从基本表元组t到单元a的向上的路径,即在立方体格中有$a\preccurlyeq t$成立,则称单元a覆盖了基本表元组t,记$BTS(a)$为a所覆盖的基本表元组集.

若两个单元a和b所覆盖的元组集是相同的,即$BTS(a)=BTS(b)$,则称a和b是覆盖等价的,记为$a\equiv_{cov}b$.

例2. 以表1为例,单元$a=(S1,*,T1)$和单元$b=(S1,*,*)$的基本元组集均为$BST(a)=BST(b)=\{(S1,P1,T1),(S1,P2,T1)\}$,因此$a$和$b$是覆盖等价的.

定义5. 商立方体[6]. 设$<L,\preccurlyeq>$为一数据立方体,$\equiv_{cov}$为L上的覆盖等价关系.则商立方体记为$(L/\equiv_{cov},\preccurlyeq)$,使得集合$L/\equiv_{cov}$中的元素均为$L$中单元关于$\equiv_{cov}$的等价类.对于商立方体中的类$C$和$D$,当且仅当$\exists c\in C,\exists d\in D$,使得$c\preccurlyeq d$成立时,才有$C\preccurlyeq D$成立.

定义6. 等价类上界(下界). 假设单元a为等价类C的上界(下界),则不存在$b\in C$,使得

$a \leqslant b(b \leqslant a)$.

例 3. 以表 1 为例，存在覆盖等价类 $C=\{(S1,*,T1),(S1,*,*),(*,*,T1)\}$，等价类中各单元的基本元组集相同，即 $BST((S1,*,T1))=BST((S1,*,*))=BST((*,*,T1))=\{(S1,P1,T1),(S1,P2,T1)\}$，并且有 $(S1,*,*)\leqslant(S1,*,T1)$，$(*,*,T1)\leqslant(S1,*,T1)$ 成立，C 中不存在单元 a，使得 $(S1,*,T1)\leqslant a$，因此 $(S1,*,T1)$ 为 C 的上界.

引理 1. 覆盖划分[6]. 覆盖等价所诱导的划分是凸的，即一个覆盖等价类 C 中，对于任意单元 $a,b\in C$，不存在单元 $c\notin C$，使得 $a\prec c\prec b$. 覆盖等价单元在任意度量上的任意聚集值都相同，并且在覆盖划分中的每个等价类都具有一个唯一的上界.

文献[9]指出只存储各个覆盖等价类上界单元的商立方体，能够在不损失原立方体语义信息的情况下缩减存储空间的开销. 根据引理 1 和覆盖等价的定义可知，单元的覆盖等价关系可以推导出它们关于所有聚集函数（如 SUM、COUNT、MIN、MAX 等）的等价关系. 同时，使用覆盖等价对数据立方体单元进行划分，不仅能够有效地压缩立方体体积并保留立方体语义，还能够支持不同类型的 OLAP 查询[9]（点查询、范围查询以及冰山查询）.

定义 7. 封闭元组[17]. 对于单元 $a\in L$，如果 $\nexists b\in L$，使得 $b\neq a$，a 覆盖 b，且 $BTS(a)=BTS(b)$，则称元组 a 为封闭元组.

定义 8. 封闭数据立方体[17]. $L_{close}=\{a|a\in L,$ a 为封闭元组$\}$.

由封闭元组的定义可知，封闭元组等价于商立方体中的等价类上界单元，因此封闭立方体实际上为仅保留等价类上界单元的商立方体. 而与文献[9]提出的树形结构不同的是，封闭立方体用关系作为存储结构来存储等价类上界单元.

3.2 频繁项集挖掘

设 $\mathcal{I}=\{i_1,i_2,\cdots,i_n\}$ 为项的全集，设数据库中事务的集合为 $TD=(Tid,\mathcal{I})$，每个事务 I 为一个非空项集，且 $I\subseteq\mathcal{I}$，每个事务都有一个标识符 $tid\in Tid$. 对于一个项集 I，记其所对应的 tid 的集合为 $t(I)$[16]（即所有包含 I 作为子集的事务 tid 的集合）. 对于一个事务标识符集合 $tidset$，记其所对应的项集为 $i(tidset)$（即 $tidset$ 中所有事务的共有项的集合）.

项集 I 的支持度 $supp(I)$ 等于它作为子集在事务中出现的次数，因此有 $supp(I)=|t(I)|$. 当项集 I 的支持度大于等于用户指定的最小支持度阈值 min_sup 时，即 $supp(I)\geqslant min\_sup$，称项集 I 是频繁的. 若对于项集 I，不存在与 I 具有相同支持度计数的项集 J，使得 $I\subset J$，则称 I 为闭项集. 根据闭项集的定义，若项集 I 为闭项集，则 I 满足 $i(t(I))=I$.

对满足最小支持度的项集构造格结构，以二元组 $(tidset,I)$ 作为节点，其中 $tidset=t(I)$. 假设二元组 $c_1(t(I_1),I_1)$ 和 $c_2(t(I_2),I_2)$ 为项集格 $<\mathcal{I},\leqslant>$ 中的两个节点，当且仅当 $I_1\subset I_2(t(I_2)\subset t(I_1))$ 成立时，有 $c_1\leqslant c_2$. 称 c_1 为 c_2 的后代，c_2 为 c_1 的祖先. 特别地，若 $|I_1|=|I_2|-1$，则称 c_1 为 c_2 的孩子，c_2 为 c_1 的双亲.

性质 1. 闭项集格节点及其偏序关系. 如果 $<C,\leqslant>$ 为一个闭项集格，则 $\mathcal{C}$ 中每个节点 c 均满足 $c(tidset,I)=c(t(I),i(t(I)))$. 对于任意节点 $c_1(t(I_1),I_1),c_2(t(I_2),I_2)\in\mathcal{C}$，如果 $c_1\leqslant c_2$，则有 $I_1\subset I_2(t(I_2)\subset t(I_1))$ 成立.

证明. 给定 $c\in\mathcal{C}$ 为格中节点，则 $c(tidset_c,I_c)=c(t(I_c),I_c)$ 中有 I_c 为一闭项集. 根据闭项集的定义可知 I_c 满足 $i(t(I_c))=I_c$，故 $c(t(I_c),I_c)=c(t(I_c),i(t(I_c)))$. 设 $<\mathcal{I},\leqslant>$ 为相应的项集格，易知 $\mathcal{C}\subset\mathcal{I}$. 如果 $c_1\leqslant c_2$ 在闭项集格中成立，则在项集格中也一定成立，根据项集格中偏序关系的定义，此时有 $I_1\subset I_2(t(I_2)\subset t(I_1))$ 成立. 证毕.

例 4. 以表 2 为例，$c(\{1,2\},\{A,C\})$ 为该事务集导出的闭项集格的一个节点，对于 c 的事务标识符集合 $tidset_c=\{1,2\}$，有 $i(tidset_c)=\{A,C\}=I_c$，即事务 1 和 2 所共有的项的集合为 $\{A,C\}$，对于 c 的项集 $I_c=\{A,C\}$，有 $t(I_c)=\{1,2\}=tidset_c$，即事务集中包含项集 I_c 的事务 tid 为 1 和 2. 此时有 $I_c=i(tidset_c)=i(t(I_c))$. 对于闭项集节点 $c_1(\{1\},\{A,B,C\})$ 和 $c_2(\{1,2\},\{A,C\})$，显然它们也属于相应的项集格. 因此，若 $c_1\leqslant c_2$ 成立，则有 $\{1\}\subseteq\{1,2\}(\{A,C\}\subseteq\{A,B,C\})$ 成立.

4 统一计算模型

本节使用集合与代数格等基本概念，形式化地论证数据立方体计算与频繁项集挖掘之间的统一关系并给出二者的映射关系.

4.1 数据立方体基本表与频繁项集挖掘事务集的映射

为便于后续关于数据立方体与频繁项集格映射的讨论，对数据立方体基本表与频繁项集挖掘事务集进行映射.

定义9. 基本表与事务集的映射函数. 令数据库基本表$R=(D,M)$,其中$D=\{D_1,D_2,\cdots,D_n\}$为基本表R的维度属性集,$M=\{M_1,M_2,\cdots,M_l\}$为基本表R的度量属性集,$t=(d_1,d_2,\cdots,d_n,m_1,m_2,\cdots,m_l)$为$R$上的任一数据单元. 定义基本表与事务集的映射函数$g_1(t)=\{D_i.d_i|D_i\in D,\ d_i\in t,\ i=1,\cdots,n\}$,此时$g_1(t)=\{D_1.d_1,\cdots,D_n.d_n\}$构成了事务集的一条事务.

利用定义9,可将基本表映射为事务集$TD=(Tid,\mathcal{I})$,其中Tid对应基本表中的元组id的集合,$\mathcal{I}$对应于基本表中的维度属性值的集合.

例5. 以表1给出的基本表R为例,利用定义9给出的映射函数可将该基本表映射为事务集形式,如表3. 为了书写的简洁方便,通常用表2的形式来表示R'. 具体的映射可以理解为:$S.S1\Rightarrow A$, $P.P1\Rightarrow B$, $T.T1\Rightarrow C$, $P.P2\Rightarrow D$, $S.S2\Rightarrow E$, $T.T2\Rightarrow F$.

表3 基本表R映射后的事务集形式R'

S	P	T	$sales$
$S.S1$	$P.P1$	$T.T1$	3
$S.S1$	$P.P2$	$T.T1$	6
$S.S2$	$P.P1$	$T.T2$	9

映射函数$g_1(t)$本质上是以不同属性为命名空间,将基本表中不同属性的属性值映射至同一域.

现对事务集向基本表进行映射. 由于事务集中每个事务的项集可以不是等长的,因此在事务集与基本表的映射中,可以分为等列事务集和不等列事务集两种情况. 定义10和定义11分别给出了等列数据集和不等列事务集与基本表之间的映射关系.

定义10. 等列事务集与基本表的映射函数. 令$TD=(Tid,\mathcal{I})$为事务集,$tr=(tid,I)$为事务集TD的任一事务,$I=\{itemid_1,\cdots,itemid_n\}\subseteq\mathcal{I}$. 定义事务集与基本表的映射函数$g_2(tr)=\{tid,d_1,\cdots,d_n|d_1=itemid_1,\cdots,d_n=itemid_n\}$,此时$g_2(tr)=\{tid,d_1,\cdots,d_n\}$构成基本表的一条元组.

定义11. 不等列事务集与基本表的映射函数. 令$TD=(Tid,\mathcal{I})$为事务集,$tr=(tid,I)$为事务集TD的任一事务,$I=\{itemid_1,\cdots,itemid_k\}\subseteq\mathcal{I}$. 定义事务集与基本表的映射函数$g_3(tr)=\{(tid,itemid_j)|j=1,\cdots,k\}$,此时$g_3(tr)=\{(tid,itemid_1),\cdots,(tid,itemid_k)\}$构成基本表中对应$tid$的$k$条元组.

通过映射函数,频繁项集挖掘的事务集中特定的事务项都可以视为数据立方体的基本表中的属性值,从而建立频繁项集挖掘与数据立方体计算的联系.

定理1. 数据立方体格$<L,\leqslant>$与频繁项集格$<\mathcal{I},\leqslant>$同构.

证明. 对于给定的基本表R,根据定义9可以将其转换为事务集R'. 设$<L,\leqslant>$为R导出的数据立方体,$<\mathcal{I},\leqslant>$为R'导出的项集格. $\forall c_1(x_1,x_2,\cdots,x_n),c_2(y_1,y_2,\cdots,y_n)\in L$,令$c_1,c_2$的映射项集为$I_{c_1},I_{c_2}$,其中映射项集$I_{c_i}$表示$c_i$中取值不为$ALL$的维值的集合. 如果$c_1\leqslant c_2$成立,根据数据立方体的定义,每当$x_i\neq ALL$时,都有$y_i=x_i$成立,因此有$I_{c_1}\subset I_{c_2}$成立. 此时,$\exists I_{c_1'}=(D_1{\cdot}x_1,D_2{\cdot}x_2,\cdots,D_n{\cdot}x_n)$,$I_{c_2'}=(D_1\cdot y_1,D_2\cdot y_2,\cdots,D_n\cdot y_n)\in\mathcal{I}$,由于$I_{c_1}\subset I_{c_2}$成立,故有$I_{c_1'}\subset I_{c_2'}$成立. 由项集格的偏序关系的定义可知,$c_1'(t(I_{c_1'}),I_{c_1'})\leqslant c_2'(t(I_{c_2'}),I_{c_2'})$成立. 显然,数据立方体中$k$维方体对应于频繁项集格中$k$项集的计算,因此由基本表导出的数据立方体与项集格的节点与节点间的偏序关系具有一一映射关系. 同理,根据定义10和定义11,将事务集转换为相应的基本表,可证得由事务集导出的项集格与数据立方体的节点与节点间偏序关系具有一一映射关系. 因此,针对同一数据集,其导出的数据立方体$<L,\leqslant>$与项集格$<\mathcal{I},\leqslant>$同构. 证毕.

定理1证明了数据立方体格和频繁项集格的节点(即相应数据单元和项集)一一映射关系以及边(偏序关系)的一一映射关系,从而为以下数据立方体算法和频繁项集挖掘算法的映射提供了一定依据.

4.2 数据立方体计算与频繁项集挖掘的映射

本节将论证数据立方体的经典计算方法与频繁项集挖掘过程的映射关系,表4简要介绍了二者的一系列算法以及基本对应关系.

4.2.1 冰山立方体计算与频繁项集挖掘的映射

定理2. 冰山立方体满足HAVING COUNT(*)$\geqslant min_sup$的数据单元等价于满足$supp(I)\geqslant min_sup$的频繁项集.

频繁项集挖掘中,用户可以指定最小支持度阈值min_sup. 频繁项集定义为满足支持度大于min_sup的项集,即:

$$supp(I)=\mathrm{COUNT}(*)\geqslant min\_sup$$

如果冰山立方体计算中,HAVING子句中使用的冰山度量为COUNT,则取相同的最小支持度阈值min_sup时,冰山立方体与频繁项集分别为在原

表4　数据立方体计算与频繁项集挖掘的映射

数据立方体计算算法	数据立方体与项集格的映射关系	频繁项集挖掘算法
完全数据立方体	项集格	Apriori系列(Apriori[10]、AprioriTID[10]、DIC[45]、Partition[46])、FP-Growth[11]、Eclat[12]、LCM[13-15]
最小支持度阈值为*min_sup*的冰山立方体(BUC[8]、H-Cubing[41]、Star-Cubing[42]、MM-Cubing[43])	最小支持度阈值为*min_sup*的频繁项集格	
浓缩立方体[4]	*min_sup*＝2的频繁项集格	
商立方体/封闭立方体(QC[6]、QC-Tree[9]、C-Cubing[44])	闭项集格	A-close[47]、CLOSET[48]、FPClose[49]、LCM[13-15]、DCI Closed[50]

始的数据立方体或频繁项集上进行剪枝后所得的结果.因此,针对同一数据集和最小支持度阈值的条件下,冰山立方体与频繁项集的计算结果相一致.

4.2.2　浓缩立方体计算与频繁项集挖掘的映射

定理3.　浓缩立方体中除BST以外的数据单元等价于*min_sup*＝2的频繁项集.

证明(反证法).　假设浓缩立方体L_c中,除了BST以外的数据单元中,存在支持度小于2的频繁项集(单元),即$\exists c\in L_c$,使得COUNT$(I_c)=1$.由于c不为BST,且COUNT$(I_c)=1$表示属性值集合I_c在基本表元组中仅出现了一次,因此在单元c包含的I_c上进行划分时只包含一个基本表元组.根据BST的定义(定义2)可知,单元c可以基于BST生成,与浓缩立方体的定义相悖.故浓缩立方体中,除BST以外的数据单元等价于支持度大于等于2的频繁项集.　　证毕.

例6.　取聚集函数为COUNT,表1导出的完全立方体如表5.以基本表元组$(S1,P2,T1,12)$为例,存在四个数据单元仅由该元组聚集计算所得:$(*,P2,*,12),(S1,P2,*,12),(*,P2,T1,12),(S1,P2,T1,12)$.因此这四个单元可以浓缩至基本单元组$(S1,P2,T1,12)$中.浓缩立方体保存了BST和在某属性(组)上进行划分时包含多条基本表元组的单元,而其他的单元可以根据它们之间的关系推算出来,根据定义3计算所得的浓缩立方体如表6所示.

根据表1导出的浓缩立方体如图2,其中每个虚线圈为一个浓缩立方体的划分.而相应的频繁项集格如图3所示,虚线圈中为事务集元组和支持度大于等于2的项集.显然浓缩立方体与$supp(I)\geqslant 2$的频繁项集具有相同的偏序关系.

4.2.3　商立方体计算与频繁闭项集挖掘的映射

定理4.　商立方体的等价类上界等价于闭项集,即商立方体等价于*min_sup*＝1的频繁闭项集(即闭项集格).

表5　完全立方体

方体	*S*	*P*	*T*	*M*
ALL	*	*	*	3
SPT	*S*1	*P*1	*T*1	1
SP	*S*1	*P*1	*	1
PT	*	*P*1	*T*1	1
SPT	*S*1	*P*2	*T*1	1
P	*	*P*2	*	1
SP	*S*1	*P*2	*	1
PT	*	*P*2	*T*1	1
SPT	*S*2	*P*1	*T*2	1
S	*S*2	*	*	1
T	*	*	*T*2	1
SP	*S*2	*P*1	*	1
ST	*S*2	*	*T*2	1
PT	*	*P*1	*T*2	1
S	*S*1	*	*	2
P	*	*P*1	*	2
T	*	*	*T*1	2
ST	*S*1	*	*T*1	2

表6　浓缩立方体

S	*P*	*T*	*M*
*S*1	*P*1	*T*1	1
*S*1	*P*2	*T*1	1
*S*2	*P*1	*T*2	1
*S*1	*	*	2
*	*P*1	*	2
*	*	*T*1	2
*S*1	*	*T*1	2
*	*	*	3

证明.　设$c_1(t(I_{c_1}),I_{c_1}),c_2(t(I_{c_2}),I_{c_2})\in QC$为商立方体$QC$中的两个等价类上界.根据$t(I)$的定义可知,$t(I_{c_1})$表示事务集中所有包含项集$I_{c_1}$的事务标识符集合,而$BTS(c_1)$表示单元$c_1$所覆盖的所有基本表元组,故此时$t(I_{c_1})=BTS(c_1)$.由商立方体的定义可知,不存在$\exists e\in QC$,使得$I_{c_1}\subset I_e$且$t(I_e)=$

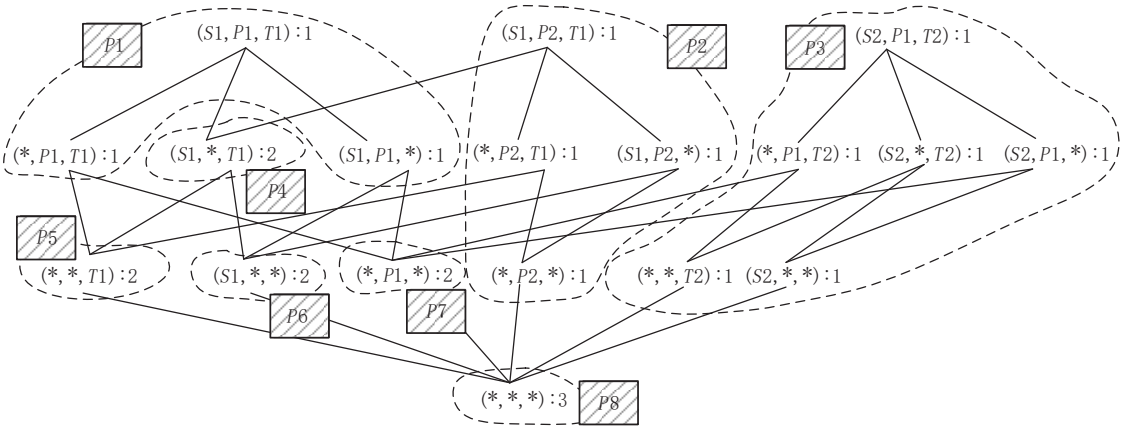


图2 浓缩立方体

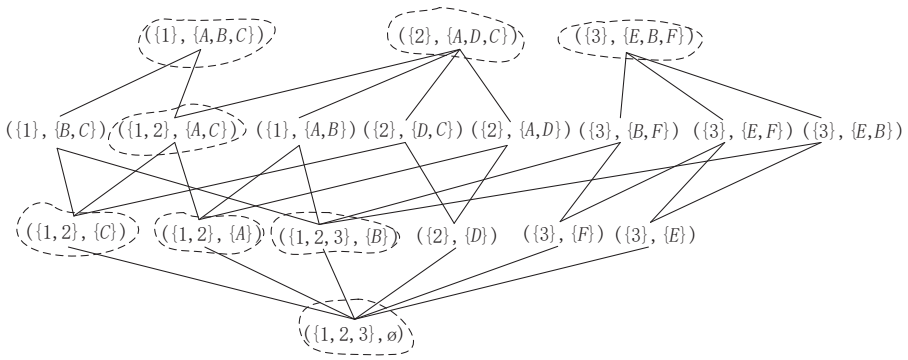


图3 *min_sup*＝2的频繁项集

$t(I_{c_1})$, 即 $c_1(t(I_{c_1}), I_{c_1}) = c_1(t(I_{c_1}), i(t(I_{c_1})))$, 故 I_{c_1} 为一个闭项集,商立方体中的所有等价类上界的维度属性集合均为项集$\mathcal{I}$(项集$\mathcal{I}$由所有维的所有维值构成)的一个闭项集. 根据频繁项集中节点间的偏序关系定义,可证得$c_1 \leqslant c_2$当且仅当$I_{c_1} \subset I_{c_2}$,与商立方体中等价类上界的偏序关系相一致. 证毕.

综上所述,频繁闭项集与商立方体具有相同的节点及节点间的偏序关系,故频繁闭项集与商立方体能够相互映射.

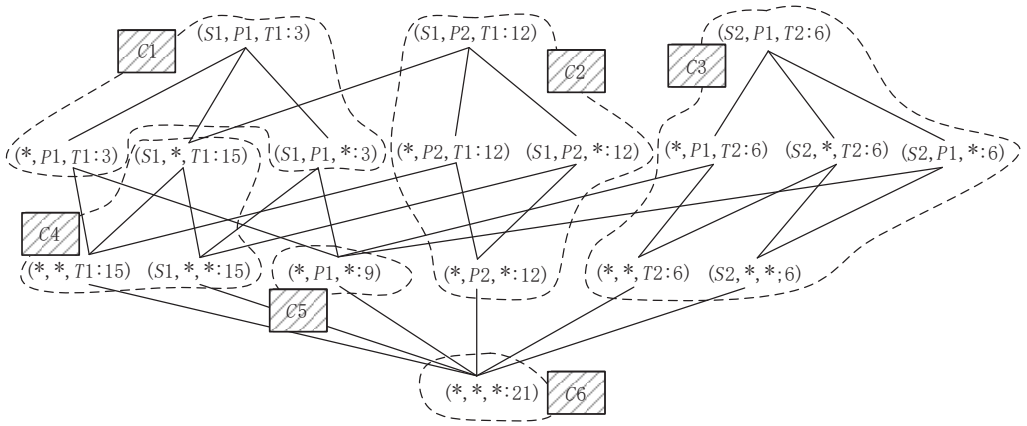
例7. 例如表1所示,基本表包含三个元组:($S1$, $P1$, $T1$)、($S1$, $P2$, $T1$)和($S2$, $P1$, $T2$). 对基本表的数据立方体计算可生成18个数据单元,根据计算依赖性形成图4(a)的数据立方体. 数据立方体根据等价关系共划分为6个等价类$C1$至$C6$,形成商立方体结构. 对表2的频繁项集挖掘可生成项集间双亲-孩子关系如图4(b),虚线圈对具有相同*tidset*的项集进行划分,得到频繁闭项集格结构. 图5(a)为表1导出的仅保留等价类上界单元的商立方体,而图5(b)为基本表R(表1)映射为事务集形式R'(表2)后计算所得的频繁闭项集. 可见在同一数据集上做商立方体计算与闭项集挖掘将产生相同结果.

4.2.4 频繁项集挖掘的泛化

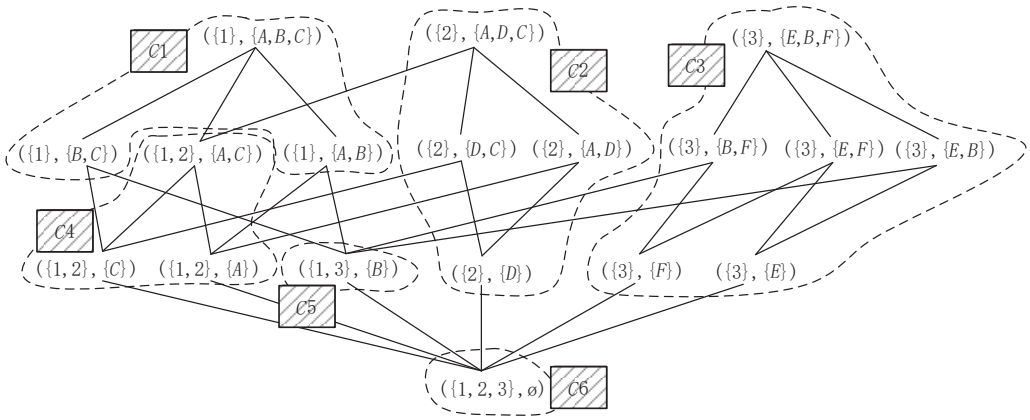
在数据立方体计算中,可以配合HAVING算子对立方体单元进行剪枝. 大多数冰山条件所使用的度量函数为COUNT,但为了考虑更广泛的语义关系,冰山度量也可以取SUM、AVG等更复杂的度量,从而得到满足不同用户需求的数据单元. 进行关联规则挖掘时,传统支持度仅使用项集的计数值来衡量项集的“频繁”与否,但在OLAP中,用户通常更倾向于根据度量的不同聚集值来分析事实,这比仅使用计数值更具说服力. 为此,我们在文献[38]提出的基于求和函数的泛化支持度和泛化置信度的基础上,给出基于任意聚集函数的泛化支持度和泛化置信度的定义,以考虑更多用户可能指定的不同聚集函数的情况.

定义12. 泛化项集. 将加入度量属性M的项集称为泛化项集,即泛化项集定义为$I_G = \{i_1, i_2, \cdots, i_n, M\}$.

定义13. 泛化支持度和泛化置信度. 对于定义在泛化项集I_G上的关联规则$A \Rightarrow B$,即$A = \{x_1, \cdots, x_s\} \subseteq I_G$, $B = \{y_1, \cdots, y_t\} \subseteq I_G$,定义该规则

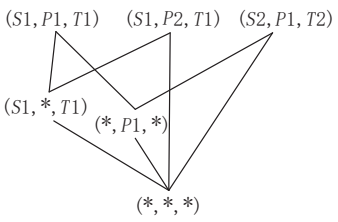


(a) 商立方体计算

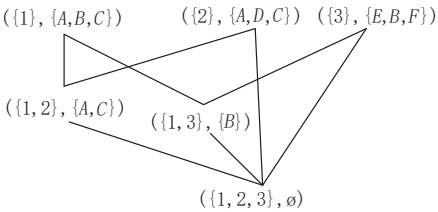


(b) 闭项集计算

图4　商立方体和闭项集



(a) 商立方体



(b) 闭项集格

图5　商立方体等价类上界单元与闭项集

的泛化支持度和泛化置信度如下：

$$supp_G(A\Rightarrow B)=\frac{f(A\cup B)}{f(I_G)}=\frac{f(x_1,\cdots,x_s,y_1,\cdots,y_t)}{f(I_G)}$$

$$conf_G(A\Rightarrow B)=\frac{f(A\cup B)}{f(A)}=\frac{f(x_1,\cdots,x_s,y_1,\cdots,y_t)}{f(x_1,\cdots,x_s)}$$

其中f为用户指定的聚集函数，可以为COUNT、SUM、AVG、MAX等. 显然，当聚集函数f取COUNT时，泛化支持度和泛化置信度等同于传统定义的支持度和置信度. 利用泛化支持度的定义来计算泛化频繁项集，当项集I满足$supp_G(I)=f(I)\geqslant min_sup$时，称$I$为基于聚集函数$f$的泛化频繁项集.

性质2.　反单调性.　如果某个泛化项集X违反某条件(例如，HAVING子句指定的冰山条件)，则X的每个泛化超集$Y\supset X$也将违反该条件. 遵守这一性质的度量称为反单调的.

频繁项集挖掘和数据立方体计算是在不同时期从不同角度提出的，其侧重点不一样. 频繁项集挖掘查找(同一属性)数据间的关联关系或相关关系，数据立方体是(不同属性)数据不同粒度的聚合或综合. 而由上可得，频繁项集挖掘的事务集加上度量属性和度量聚集函数即可视为数据立方体计算，其支持度、置信度等只是某种度量聚集函数. 因此，一定程度上，频繁项集挖掘可视为数据立方体计算的特例.

5 统一计算算法示例

为了验证数据立方体计算与频繁项集挖掘的映射和融合，本节首先将频繁项集挖掘的事务集映射到关系数据库的基本表中，将频繁项集计算映射为冰山立方体计算，使得在关系数据库中就能够用标准的SQL-92或扩展的SQL进行频繁项集挖掘，从而避免可能不必要的数据移动；同时，向事务集中加入度量属性，构成泛化事务集，将在泛化事务集上的频繁项集计算映射为冰山立方体计算；然后，将基本表与事务集进行映射，商立方体计算映射为频繁项集挖掘中的闭项集计算，以提升数据立方体的计算效率.

5.1 频繁项集挖掘与冰山立方体计算的融合

根据实际应用情况，将事务集分为了等列事务集和不等列事务集两种类型，并映射为基本表的形式. 再使用冰山立方体计算的SQL语句实现不同类型的事务集上的频繁项集挖掘. 最后，加入度量属性构成泛化项集，利用泛化支持度做频繁项集挖掘，在等列数据集上聚集函数取SUM和MAX时，频繁项集挖掘与冰山立方体计算的统一性.

5.1.1 等列事务集的冰山立方体计算

给定等列事务集$TD=(Tid, \mathcal{I})$，根据定义10可以将其映射为基本表$R=(Tid, D_1, \cdots, D_n)$. 对于用户指定的支持度阈值$min_sup$，可以使用冰山立方体的计算方法(WITH CUBE算子与HAVING算子的结合)来计算频繁项集. WITH CUBE算子自动对GROUP BY子句中列出的维度属性集合求幂集，即给出所有维度属性组合；如果SQL不支持WITH CUBE算子，可以将WITH CUBE改写为GROUP BY不同属性组合. HAVING算子对每个维组合判断其是否满足用户指定阈值.

由第三节的论述可知，当聚集函数为COUNT时，对于同一支持度阈值min_sup，在数据集上计算得到的冰山立方体与频繁项集相互等价. 因此，可以利用SQL语句实现事务集的频繁项集挖掘，相应的SQL语句如下：

SELECT $D_1,\cdots,D_n$, COUNT(*)
FROM R
GROUP BY $D_1,\cdots,D_n$ WITH CUBE
HAVING COUNT(*) >= min_sup;

例8. 以表2所示事务集为例，其可视为等列基本表，在其上使用WITH CUBE算子进行查询，并取最小支持度阈值$min_sup=2$，相应SQL语句如下：

SELECT $S, P, T,$ COUNT(*)
FROM R'
GROUP BY S, P, T WITH CUBE
HAVING COUNT(*) >= 2;

执行SQL语句，得到查询结果如表7所示. 其中，维值为NULL表示维值取ALL. SQL语句的输出结果中，每行取值不为NULL的维值组合即为满足支持度阈值的频繁项集，COUNT列的值即为该频繁项集的支持度.

表7 事务集R'导出的冰山立方体

S	P	T	COUNT
A	NULL	C	2
A	NULL	NULL	2
NULL	NULL	C	2
NULL	B	NULL	2
NULL	NULL	NULL	3

向事务集中增加度量属性M(如购物篮分析中可以加入交易金额)，构成泛化项集. 将基于聚集函数f的泛化频繁项集挖掘映射为冰山立方体计算，相应的SQL语句如下：

SELECT $D_1,\cdots,D_n,f(M)$
FROM R
GROUP BY $D_1,\cdots,D_n$ WITH CUBE
HAVING $f(M)$ >= min_sup;

例9. 以表2所示事务集为例，增加度量属性M后如表8所示. 在其上使用WITH CUBE算子进行查询，取聚集函数为SUM和MAX，最小支持度阈值$min_sup=9$，相应SQL语句如下：

SELECT $S, P, T,$ SUM(M)
FROM R''
GROUP BY S, P, T WITH CUBE
HAVING SUM(M) >= 9;
SELECT $S, P, T,$ MAX(M)
FROM R''
GROUP BY S, P, T WITH CUBE
HAVING MAX(M) >= 9;

查询结果分别为表9和表10，SUM(MAX)列的值即为该泛化频繁项集的泛化支持度.

5.1.2 不等列事务集的冰山立方体计算

设$TD=(Tid, \mathcal{I})$为不等列事务集，利用定义11

表8　事务集R'的泛化事务集R''

tid	*S*	*P*	*T*	*M*
1	*A*	*B*	*C*	3
2	*A*	*D*	*C*	6
3	*E*	*B*	*F*	9

表9　泛化事务集R''导出的基于SUM函数的冰山立方体

S	*P*	*T*	SUM
A	NULL	*C*	9
A	NULL	NULL	9
NULL	NULL	*C*	9
NULL	*B*	NULL	12
E	*B*	*F*	9
E	*B*	NULL	9
E	NULL	*F*	9
NULL	*B*	*F*	9
E	NULL	NULL	9
NULL	NULL	*F*	9
NULL	NULL	NULL	18

表10　泛化事务集R''导出的基于MAX函数的冰山立方体

S	*P*	*T*	MAX
NULL	*B*	NULL	9
E	*B*	*F*	9
E	*B*	NULL	9
E	NULL	*F*	9
NULL	*B*	*F*	9
E	NULL	NULL	9
NULL	NULL	*F*	9
NULL	NULL	NULL	9

将其映射为基本表$R=(Tid, Itemid)$. 根据Apriori性质，将频繁项集挖掘分为两个部分：生成候选集和计算频繁k项集.

本文受文献[51]启发，利用SQL语句实现不等列事务集的频繁项集挖掘的具体步骤如下：

(1) 生成候选集C_k：利用频繁$k-1$项集来生成候选集C_k. 令L_{k-1}与其自身连接，产生候选集C_k. L_{k-1}的任一记录r_1与另一记录r_2做连接时，r_1与r_2的前$k-2$项相等，通过限制r_1的第$k-1$项小于r_2的第$k-1$项来保证不会生成重复的候选项. 具体的SQL语句如下所示：

SELECT $t_1.item_1, t_1.item_2, \cdots, t_1.item_{k-1}$,
$t_2.item_{k-1}$
INTO C_k
FROM $L_{k-1}\ t_1, L_{k-1}\ t_2$
WHERE $t_1.item_1 = t_2.item_1$ AND $\cdots$ AND $t_1.item_{k-2}=t_2.item_{k-2}$ AND $t_1.item_{k-1}<t_2.item_{k-1}$;

(2) 生成频繁k项集：对基本表做k次自连接，扫描自连接后得到的基本表，仅保留存在于候选集C_k中的记录，并计算该记录的支持度，存储满足支持度阈值的记录至L_k中. SQL语句如下所示：

SELECT $t_1.itemid, \cdots, t_k.itemid$, COUNT(*)
INTO L_k
FROM $t_1, R\ t_2, \cdots, R\ t_k$
WHERE $t_1.tid = t_2.tid$ AND $\cdots$ AND $t_{k-1}.tid = t_k.tid$
AND EXISTS (SELECT 1 FROM C_k WHERE $C_k.item_1 = t_1.itemid$ AND $\cdots$ AND $C_k.item_k = t_k.itemid$)
GROUP BY $t_1.itemid, \cdots, t_k.itemid$
HAVING COUNT(*) >= *min_sup*;

重复上述步骤直至频繁k项集为空时停止，即可得到满足最小支持度阈值的所有频繁项集. 需要注意的是，步骤(1)将生成的候选集存入临时表C_k，而不是单句SQL中将基本表不断迭代进行自连接，以至于会造成大量的临时空间.

5.2　商立方体计算与闭项集挖掘的融合

为验证数据立方体和频繁项集挖掘的统一计算方法下，能够设计出更有效的实际应用算法(如提升数据立方体的计算效率)，引入闭项集挖掘的经典算法LCM[14-15]来实现商立方体计算. 相比起冰山立方体，商立方体对原始立方体的压缩是保留语义的，当查询未保存在商立方体中的节点时，可以通过商立方体保存的上下界信息扩展得到. 但查询冰山立方体中被剪枝掉的节点时则需要进行临时计算，将产生一定的计算开销. 同时，商立方体的压缩率比起同样能保留语义的浓缩立方体而言，提升了将近三倍[6]. 因此商立方体可以视为是实现存储开销和时间开销平衡的、较优的数据立方体计算方法，若能提升商立方体的计算效率，将会有效改善OLAP效率.

传统的商立方体格构造算法(基于深度优先探索，DFS)如算法1所示，该算法可以将基本表构造成商立方体格，输出每个等价类的上界单元并统计等价类上界的个数. 首先，需要获得各个维度的属性值集合，然后从*ALL*(各个维度属性值均为"*")单元格开始(步1)，使用*DFS*函数计算出当前数据单元所属等价类的上界的度量值，依据等价类的基本元组集分解当前维度值，寻找同一维度的下一个等价类的上界，直到所有的数据单元均被搜索(步4～19). 例如针对表1，首先从*ALL*单元的第一个维度开始分解

为($S1$, *, *),($S2$, *, *)两个数据单元,寻找当前数据单元的上界($S1$, *, $T1$),($S2$, $P1$, $T2$),再从该数据单元上界的下一个维度进行分解,依照此过程,直至当前数据单元的所有维度都被分解完成.

算法1. 商立方体计算DFS[6].

输入:基本表*BaseTable*

输出:商立方体

```
1.  DFS(cell, partition, k, parentCell)
    /* cell:当前需要分解的数据单元;partition:基本表划分;k:记录数据单元分解的维度;
    parentCell:记录父亲单元 */
2.  计算cell所属等价类的上界ub;
3.  记录当前等价类的上界ub,等价类的父亲单元parentCell,上界ub的度量值m;
4.  FOR EACH 0<j<k IN cell DO
5.      IF (cell[j]==ALL && ub[j] !=ALL) DO
6.          RETURN
7.      END IF
8.  END FOR
9.  FOR EACH k<j<n IN cell DO
10.     IF (cell[j]==ALL) DO
11.         FOR each 分区partition在维度j上的值v
12.             ub[j]=v
13.             分解数据单元cell的上界ub,得到划分part
14.             IF part非空 DO
15.                 调用DFS(c,part,j,ub)
16.             END IF
17.         END FOR
18.     END IF
19. END FOR
20. RETURN
```

大部分闭项集挖掘算法在寻找闭项集时通常采用剪枝的方法来删除不重要的项集,从而枚举出来频繁闭项集.但是,这种剪枝方法是不完全的,算法会处理一些不必要的频繁项集从而产生额外的工作量.LCM算法在闭项集之间定义了一种双亲-孩子关系,这种关系诱导了仅由所有闭项集构成的树形横向路径.算法通过深度优先遍历的方式遍历所有路径,能够在多项式时间内找到每个项集的所有频繁闭项集,并且不需要额外的空间开销来存储计算过程中得到的临时闭项集.

对于项集X,索引s,$X(s)=X\cap\{1,\cdots,s\}$,则闭项集集合C中闭项集的双亲-孩子关系P的定义如下.对于项集$X\in C$,通过$P(X)=i(t(X(s(X)-1)))$来定义X的双亲,其中$s(X)$为使得$t(X)=t(X(s))$且$t(X)\neq t(X(s-1))$成立的最小项s.如果Y是X的双亲,则称X是Y的一个孩子.令$Root=i(t(\varnothing))$为C中的最小项集,称为根节点.对于任意$X\in C\backslash\{R\}$,它的双亲$P(X)$总是由C定义并属于C.

对于任意$X\in C$和它的双亲Y,有$X\subset Y$成立,因为$t(X(i(X)-1))\subset t(X)$.因此,关系P是无环的,并且其图形表示形成了一棵树.通过深度优先方法来遍历树,可以在树的大小的线性时间内枚举所有的闭项集.此外,算法不会将树存储在内存中,其从树的根$Root$开始,先找到根的孩子X,然后转到X继而寻找X的孩子.以同样的方式,当找到叶子节点时,开始回溯并找到另一个孩子.重复此过程,最终在C中找到所有的闭项集.

为了找到当前频繁项集的孩子,使用以下引理.对于项集X和索引s,令$X[s]=X\cup H$,其中H是满足项$j\geqslant s$,且$j\in i(t(X\cup\{s\}))$的集合.

引理2[14]. X'是$X\in C$的一个孩子($X'\in C$且X'的双亲为X)当且仅当:

(1)对于某些$s>s(X)$,有$X'=X[s]$;

(2)$X'=i(t(X'))$(X'是闭项集);

(3)X'是频繁项集.

根据引理2,计算闭项集的算法如算法2所示.显然,如果$X[s]=i(t(X[s]))$(条件②)成立,则$t(X[s])=t(X\cup\{s\})$成立(步2~4).注意到,不存在X的孩子X'满足$X[s]=X[s']$,$s\neq s'$,因为$X[s]\backslash X$和$X[s']\backslash X$的最小项分别为s和s'.

算法2. LCM(计算闭项集)[13-15].

输入:基本表*BaseTable*

输出:闭项集X

```
1.  FOR EACH s>s(X) DO
2.      IF X[s]是频繁的并且X[s]=i(t(X[s])) DO
3.          调用LCM_Iter(X[s])
            /*LCM_Iter()函数具体见文献[13]*/
4.      END IF
5.  END FOR
```

算法LCM以线性时间计算C中所有闭项集的数量.该算法遍历了仅由闭项集组成的树,并且每次迭代都没有以前的算法那么繁重.因此,算法在实践中运行的速度很快.注意,双亲X不是递归调用中$X[i]$的前缀,条件②的检查可以视为对非闭项集的剪枝.

4.2.3节论证了商立方体计算与频繁闭项集挖

掘的一致性，因此可以利用高效的频繁闭项集挖掘算法LCM来计算商立方体. 商立方体的DFS算法在对基本表进行商立方体构造时，输出了等价类上界单元及其个数(等价类的个数). 通过分析等价类上界和闭项集之间的关系，对基本表映射生成的事务集进行计算，映射算法如算法3所示(其中步1～2表示扫描基本表每一行的属性值，步3～5表示对该属性值加上列名或列号作为前缀). 此时闭项集挖掘算法LCM输出的闭项集与商立方体算法输出的等价类上界单元相同，并且闭项集间的偏序关系与等价类上界单元的偏序关系相同.

算法3.　映射算法.

输入：基本表*BaseTable*

输出：事务集

```
1.    FOR EACH Row IN BaseTable DO
2.      FOR EACH Dim IN Row DO
3.        IF Dim没有Pre
          /*Dim为当前扫描的值；Pre为当前值的属性(或列号)*/
4.          Dim = Pre. Dim
5.        END IF
6.      END FOR
7.    END FOR
```

例10.　以表1所示基本表R为例，利用算法3可以将其转换为如表2所示的事务集R'. 在R'上使用LCM算法进行计算，取最小支持度阈值为1，得到的结果如表11所示，而在R上利用QC算法计算所得的商立方体如表12所示. 根据基本表与事务集的转换关系可知，在R'上用闭项集挖掘算法LCM来计算商立方体所得结果与在R上用商立方体算法QC的计算结果相同.

表11　事务集R'导出的商立方体

S	P	T
*	*	*
*	B	*
A	*	C
A	B	C
A	D	C
E	B	F

6　实验分析

为了评估上述定理与融合算法的正确性以及数据立方体方法与频繁项集挖掘算法的计算效率，进行了一系列全面的实验.

表12　基本表R导出的商立方体

S	P	T
*	*	*
*	$P1$	*
$S1$	*	$T1$
$S1$	$P1$	$T1$
$S1$	$P2$	$T1$
$S2$	$P1$	$T2$

所有的实验均在以下的环境中进行：CPU为Intel Core i5-7200U@2.5 Ghz，内存为8 GB，硬盘为256 GB SSD；使用的操作系统为Ubuntu 20.04，开发语言为C++，编译环境为gcc 9.2.0.

6.1　频繁项集挖掘与冰山立方体计算的融合实验

将频繁项集挖掘事务集数据导入关系数据库(如果数据在关系数据库中则无需导入)，验证频繁项集计算与冰山立方体计算的融合，在2个真实数据集和1个合成数据集上进行实验比较，表13给出了数据集的基本信息. 其中，Mushroom①与RecordLink②为等列的真实数据集，T40I10D100K①为不等列的生成数据集.

表13　频繁项集挖掘与冰山立方体计算融合实验数据集

数据集	事务数	项数	平均事务大小	最大项集
Mushroom	8124	119	23	23
RecordLink	547913	29	10	10
T40I10D100K	100000	924	39.6	77

本节使用LCM[13-15]算法与SQL Server 2019数据库①②的WITH CUBE算子分别计算满足支持度阈值的频繁项集、泛化频繁项集和冰山立方体单元，验证两种方法融合的正确性.

关系数据库Microsoft SQL Server 2019支持WITH CUBE计算，通过联合使用WITH CUBE算子和HAVING算子进行计算可以得到满足最小支持度阈值的冰山立方体. 对于关系数据库MySQL，其支持WITH ROLLUP计算，同样可以组合为CUBE计算. 而对于SQL-92或其以上标准，可以通过GROUP BY组合为CUBE计算.

① Frequent Itemset Mining Dataset Repository, http://fimi.uantwerpen.be/data/

② SPMF: A Java Open-Source Data Mining Library, http://www.philippe-fournier-viger.com/spmf/index.php? link=datasets.php

6.1.1 基于COUNT的频繁项集与冰山立方体计算

本节令冰山立方体计算中所使用的聚集函数为COUNT，验证频繁项集挖掘与冰山立方体计算的统一性并对比二者的计算效率.

首先，保持Mushroom数据集的元组数和最小支持度阈值$min_sup=2$不变，通过投影的方法生成7个不同维度属性数（简称维数，从6增加到12）的数据集. 利用LCM和SQL Server 2019对7个数据集分别进行频繁项集计算和冰山立方体计算，计算结果如图6所示.

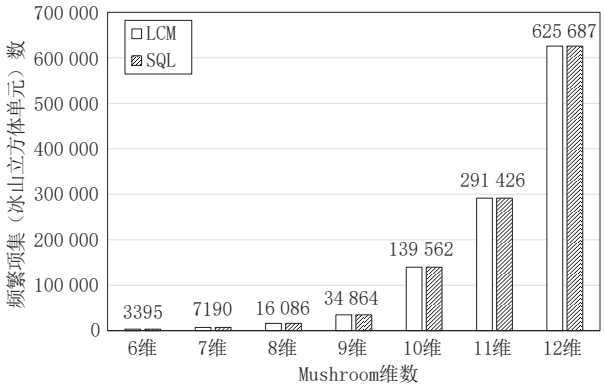


图6 Mushroom不同维数下的计算结果

其次，在RecordLink数据集上保持元组数和维数不变，最小支持度阈值在0.01%到0.1%之间变化，分别进行频繁项集计算和冰山立方体计算，结果如图7所示.

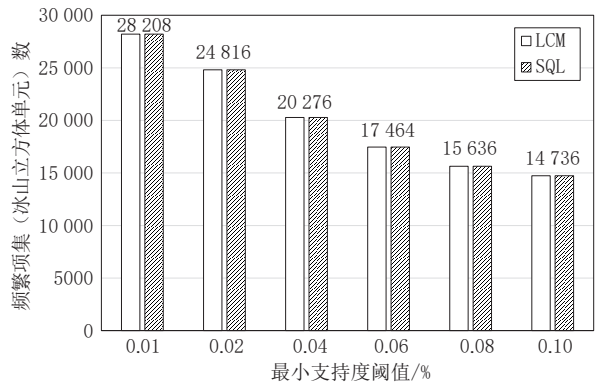


图7 RecordLink不同支持度阈值下的计算结果

最后，在不等列数据集T40I10D100K上保持元组数不变，取最小支持度阈值1.5%，对比二者计算频繁k项集的计算结果及计算效率，结果见图8.

可以发现LCM算法与SQL Server的WITH CUBE算子的计算结果为相同数量的频繁项集和冰山立方体单元. 实验还通过文本匹配的方式，对比

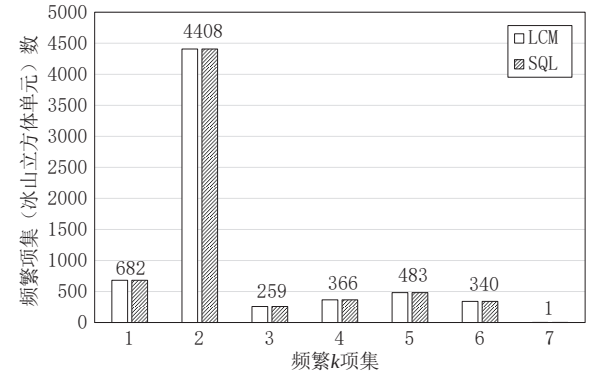


图8 T40I10D100K支持度为1.5%两种算法的频繁k项集

两种算法的输出结果，统一二者输出格式为：满足支持度阈值的维值/项的组合及其计数值. 匹配结果显示，两种算法的输出结果完全一致. 即对于同一数据集而言，计算满足用户给定的支持度阈值的频繁项集，等价于计算满足支持度阈值的冰山立方体单元.

图9和图10给出了等列数据集上两种算法的运行时间. 在支持度不变的情况下，随着维数的增加，二者的计算效率差距也在不断增加. 这是由于，在LCM算法的回溯算法中，按字典序逐个增加项来判断项是否频繁，并使用了超立方体解体（Hypercube Decomposition）的技术，将格分解为多个子格，从而减小项集的检索空间，随着维度的增加，支持度的计算仍保持在线性时间内. 并且，$k+1$项集支持度的计算过程中重用了频繁k项集的计算结果，因此支持度的变化对运行时间的影响较小.

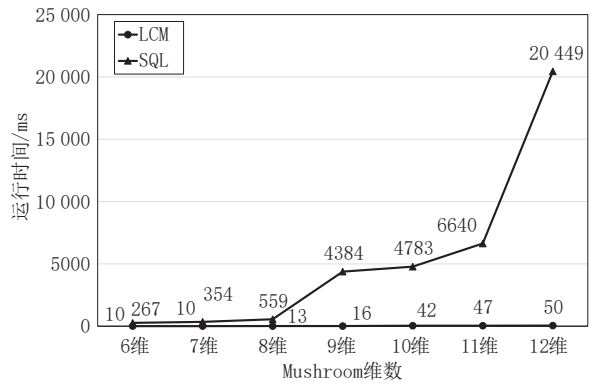


图9 Mushroom不同维数运行时间对比

在不等列数据集T40I10D100K上，最小支持度阈值设置为1.5%，此时利用SQL Server计算满足阈值的冰山立方体所需的总时间为621 040 ms，而用LCM计算满足阈值的频繁项集所需的总时间为6618 ms. 可以发现SQL Server上的运行时间要远

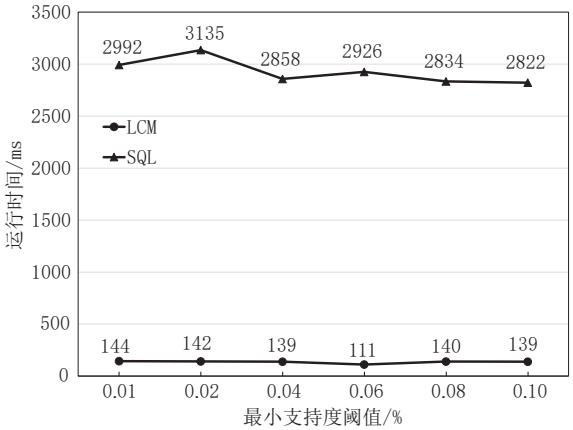


图10　RecordLink不同支持度阈值下的运行时间

大于LCM的运行时间. 这是由于，对于不等列数据而言，SQL Server上需要将其转换为单列的基本表，再通过生成候选集来计算频繁k项集，其中生成候选集时的内连接部分和生成临时表部分将产生很大的时间开销.

尽管如此，直接在数据库进行频繁项集挖掘在一些场景仍有较大意义. LCM算法前提是数据能够放入内存，并不适用于大数据场景. 近年来数据库与人工智能进行了融合，库内数据挖掘或机器学习(In-database Machine Learning)受到关注. 以上实验还表明，可以在关系数据库中通过标准的或扩展的SQL语句，对关系表进行频繁项集挖掘，从而充分利用关系数据库的特性，并避免可能向数据挖掘或机器学习平台大量的数据移动. 在下一步工作中，将尝试进一步针对数据库内频繁项集挖掘进行深度优化，如融合LCM算法思想到数据库自定义函数或直接在openGauss等开源数据库上进行优化.

6.1.2　泛化频繁项集与冰山立方体计算

为验证泛化频繁项集与冰山立方体计算映射的正确性，在等列数据集Mushroom上增加一列度量属性，其中项集度量值为0到20之间的随机整数. 实验保持元组数不变，取Mushroom中不为度量属性的6维-10维子数据集. 以聚集函数SUM和MAX为例，分别计算泛化频繁项集和冰山立方体. 其中聚集函数为SUM时，取最小支持度阈值$min_sup=1000$；聚集函数为MAX时，取最小支持度阈值$min_sup=18$. 计算结果如表14和表15所示.

实验统一二者的输出格式为由逗号连接的频繁项集和写在中括号中的泛化支持度，如表9给出的第一个冰山立方体单元$(A,*,C,9)$在结果中写为$\{A,C(9)\}$. 对比二者的输出结果发现，对于同一数据集、聚集函数和最小支持度阈值下，泛化频繁项集的计算结果与冰山立方体的计算结果完全相同. 因此，可以将冰山条件中所使用的聚集函数用于频繁项集挖掘中，根据除计数值以外的度量值来判断项集的信息价值. 由于本文所使用的泛化项集挖掘算法是以Apriori算法为基础，修改判断项集频繁与否的条件，验证泛化频繁项集与冰山立方体计算映射的正确性，计算效率有待提升，故不再给出运行效率对比. 可以根据上述泛化思想，设计出更高效、更全面的频繁项集挖掘算法.

表14　基于SUM函数的泛化频繁项集和冰山立方体计算

维数	泛化频繁项集数	冰山立方体单元数
6维	1153	1153
7维	2316	2316
8维	4779	4779
9维	9863	9863
10维	20642	20642

表15　基于MAX函数的泛化频繁项集和冰山立方体计算

维数	泛化频繁项集数	冰山立方体单元数
6维	2882	2882
7维	5982	5982
8维	13132	13132
9维	28365	28365
10维	107537	107537

6.2　浓缩立方体计算与频繁项集挖掘的融合实验

利用算法3将基本表转换为频繁项集挖掘事务集，验证浓缩立方体计算与频繁项集计算的融合. 由于篇幅有限，实验仅使用真实数据集Foodmart①来验证定理3的正确性，并对比用频繁项集挖掘算法LCM来计算浓缩立方体和用浓缩立方体经典算法Min_Cube[4]进行计算的计算效率.

实验保持Foodmart数据集维数不变，元组数从1000增加至13 795. 实验的计算结果如表16所示.

可以发现，LCM算法计算所得的$supp\geqslant 2$的频繁项集数等于Min_Cube算法计算所得的浓缩立方体单元数减去基本单元组数. 输出结果对比显示，Min_Cube计算所得的除基本单元组以外的浓缩立方体单元及其COUNT值，与LCM计算所得的$min_sup=2$的频繁项集及其支持度完全一致. 以表1和表2为例，LCM的计算结果如表17所示，

① Microsoft SQL Server 2000 Analysis Services提供

表16 Foodmart不同元组数下计算结果

元组数	LCM(支持度大于等于2的项集数)	Min_Cube(浓缩立方体单元数)	BST(基本单元组数)
1000	232 728	233 728	1000
4000	853 630	857 630	4000
8000	1 724 544	1 732 544	8000
12000	2 584 530	2 596 530	12 000
13795	2 934 116	2 947 911	13 795

Min_Cube的计算结果如表18所示.

表17 LCM在事务集R'上的计算结果

S	P	T	M
A	*	*	2
*	B	*	2
*	*	C	2
A	*	C	2
*	*	*	3

表18 Min_Cube在基本表R上的计算结果

S	P	T	M
$S1$	$P1$	$T1$	1
$S1$	$P2$	$T1$	1
$S2$	$P1$	$T2$	1
$S1$	*	*	2
*	$P1$	*	2
*	*	$T1$	2
$S1$	*	$T1$	2
*	*	*	3

算法运行时间如图11所示.Min_Cube算法中需要产生不同维度属性值的组合,并为每个维度属性组合生成候选元组位图索引和方体的位图索引来判断BST,产生了较大的空间开销和时间开销.

6.3 商立方体计算与闭项集挖掘的融合实验

将基本表转换为频繁项集挖掘事务集,验证商立方体计算与闭项集计算的融合.实验同时使用了真实数据集和合成数据集来评估融合算法的时间效率,数据集的信息如表19所示.

表19 商立方体计算与闭项集挖掘融合实验数据集

数据集	元组数	维数
Foodmart	13 795	11维
Weather[52]	500 000	11维
符合Zipf分布的合成数据集①	100 000	10维

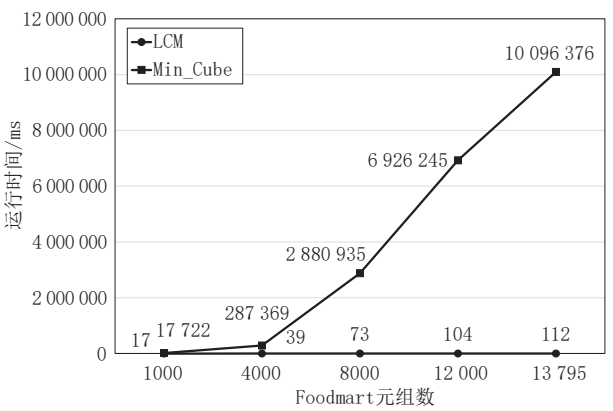


图11 Foodmart不同元组数运行时间对比

真实数据集Foodmart和Weather均常用于验证各种数据立方体方法的有效性.而利用Zipf分布生成的合成数据是一种标准数据集,常用于测试不同条件下算法的性能.

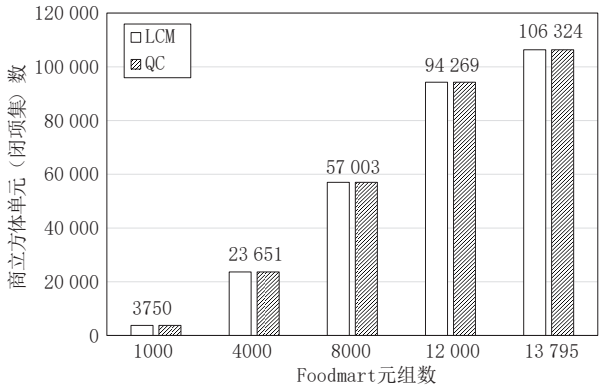
本节实验比较的算法为:QC[6],数据立方体计算较优的算法;LCM[13-15],频繁项集挖掘较优的算法.在相应数据集上进行了多次且全面的测试,每次测试重复3次.采用算法3将基本表映射为事务集,实际上,基本表与事务集的映射算法具有线性时间复杂度,在总计算开销中占比较小,对大数据集而言可以忽略不计.为了简短起见,在下面只给出一些具有代表性的结果.

6.3.1 算法正确性实验

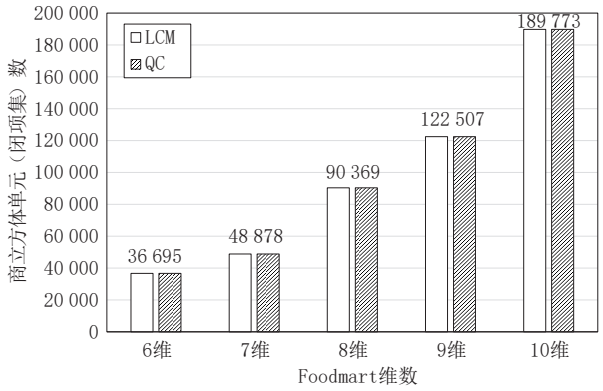
为了验证商立方体计算与闭项集计算之间映射的正确性,在Foodmart数据集上分别改变元组数和维度来测试二者计算出的结果是否一致,即对比利用LCM计算出的数据集的闭项集,和QC计算出的数据集的商立方体单元.首先,改变Foodmart数据集的元组数来进行测试,其中元组数从1000增加到13 795.然后,保持元组数不变,改变数据集的维度来进行测试,维度是从原本的11个维度中随机抽取6-10个维度进行组合,形成新的数据集.两种算法的计算结果如图12所示.

在输出结果的核对上,以表1的输出为例.表1的3条元组($S1$, $P1$, $T1$)、($S1$, $P2$, $T1$)和($S2$, $P1$, $T2$)在LCM算法的输出为{}、{$P1$}和{$S1$, $T1$}({}代表数据立方体里的(*, *, *)单元);在QC算法里的相

① http://jimgray.azurewebsites.net/dbgen/

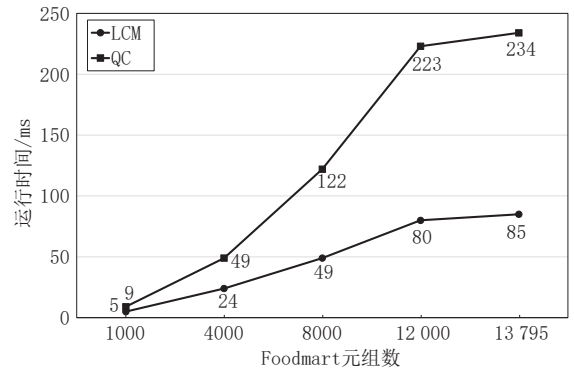


（a）不同元组数计算结果对比

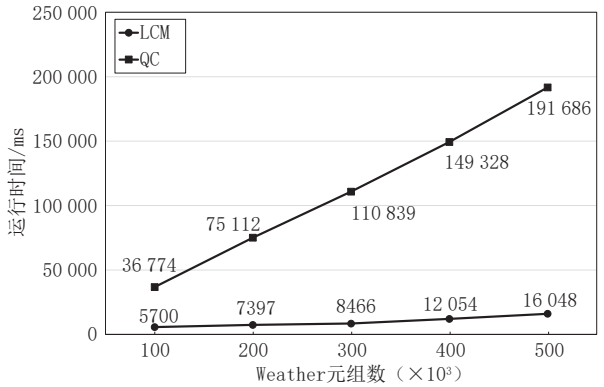


（b）不同维数计算结果对比

图12　Foodmart不同元组数和维数两种算法分别计算结果



（a）Foodmart数据集上运行时间对比



（b）Weather数据集上运行时间对比

图13　Foodmart和Weather不同元组数运行时间对比

应输出为(*, *, *)、(*, $P1$, *)和($S1$, *, $T1$). 实验通过统一两种算法的输出结果格式，对Foodmart数据集上两种算法计算所得的结果进行了文本对比. 输出结果对比表明，两种算法的计算结果完全一致，进一步验证了融合算法的正确性. 通过上述的两个实验，证明了二者之间映射的正确性. 后续实验主要进行性能比较.

6.3.2　不同元组数对比

分别在Foodmart和Weather数据集上测试上述两个算法，保持数据集维度不变，变化元组数进行对比. 实验分别用关联规则挖掘的LCM算法计算映射后的Foodmart和Weather数据集的闭项集及其偏序关系，以及用商立方体的QC算法计算原始数据集中等价类上界及其偏序关系，发现LCM计算出的闭项集和QC计算出的商立方体相一致，且计算所得的格结构（节点间的双亲-孩子关系）相同. 统计在不同元组数的情况下两种算法的运行时间，结果如图13所示.

从图13可以发现，当数据集元组较少的时候，两种算法的效率相差不大，但随着元组数的增加，二者的效率相差越来越大，当Foodmart数据集的元组数达到12 000行时，LCM算法的效率几乎提高了3倍. 同样，在Weather数据集上测试结果也是随着元组数的增加，二者的效率差距越来越明显. 这是因为相较于QC算法，LCM算法在剪枝的过程中效率更高. LCM算法通过重用闭算子的计算结果和生成检索子表的方式，大大缩小了生成闭项集（商立方体单元）时的探索空间，同时避免了QC算法中临时类的生成与合并的过程，有效减少了计算量.

相对于商立方体的QC算法，闭项集挖掘LCM算法在两个数据集上的效率提升分别如图14和图15所示. 当Foodmart元组数越来越多时，LCM算法运行的时间效率会提升越来越高. 而在Weather数据集中，当元组数到达一定数量时，提升效果保持稳定，高达92%，QC的运行时间远远大于LCM的运行时间. 这是由于LCM的空间复杂度和时间复杂度均为线性复杂度，并做了多种优化组合，包括扩展了前缀保留闭项集[14]（Prefix Preserving Closure，PPC），减少了剪枝的时间，所以在元组数越多的时候，其效率越高.

6.3.3　不同维数对比

本实验在保持Foodmart和Weather数据集元

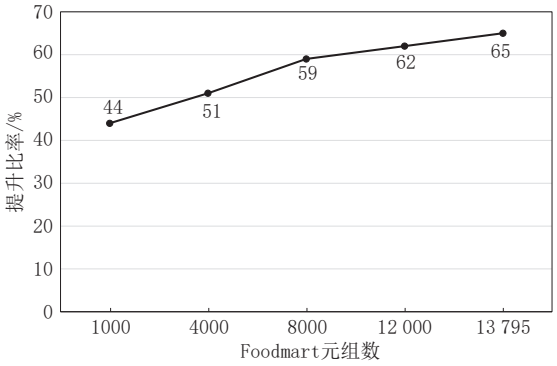


图14 Foodmart数据集上时间效率的提升

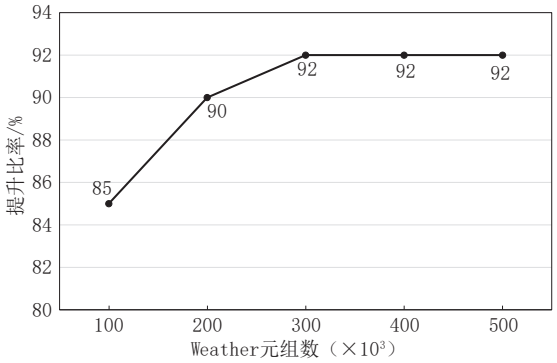


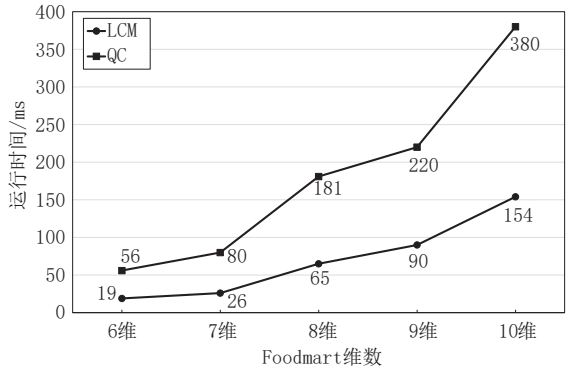
图15 Weather数据集不同元组数时间效率的提升

组数不变的情况下，通过改变维度来进行对比. 分别在两个数据集上，从11个维度中随机抽取了6、7、8、9、10个维度进行对比实验，结果如图16所示. 可以看出，在元组数不变的情况下，维度数越多，二者的运行效率差距越大. 特别是在10维的Weather数据集上，QC的运行时间将近为LCM的13倍.

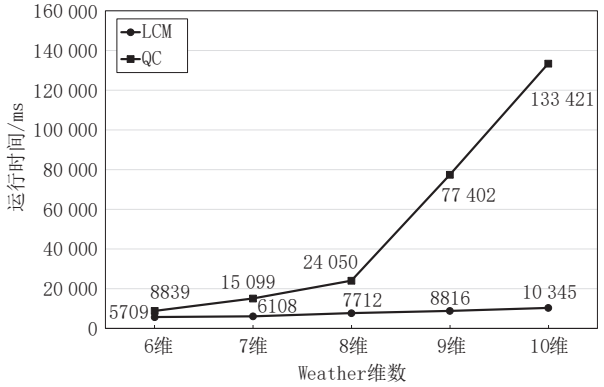
根据不同维度的运行时间，可以得到在维度数改变的情况下，算法运行效率的变化. 效率变化统计如图17所示. 当元组数保持不变时，随着维数的增长，LCM算法在两个数据集上的计算效率不断提升. 这是由于LCM算法遍历只由闭项集组成的树，并进行了有效剪枝，所以维度越多时，其效率越高.

6.3.4 不同倾斜度对比

本实验通过改变Zipf因子来改变合成数据集的数据倾斜度，比较了数据倾斜度对算法计算效率的影响. 为了测量数据倾斜度的影响，将维数固定为10，将元组的数量固定为10^5，Zipf因子在0到0.5之间变化，统计在不同倾斜度下两种算法的运行时间. Zipf因子越大，数据倾斜程度越大，即某些维值出现的频率越高，每个维出现的不同维值的数量越少[17]. 算法运行时间对比如图18所示.



（a）Foodmart运行时间对比



（b）Weather运行时间对比

图16 Foodmart和Weather不同维度运行时间对比

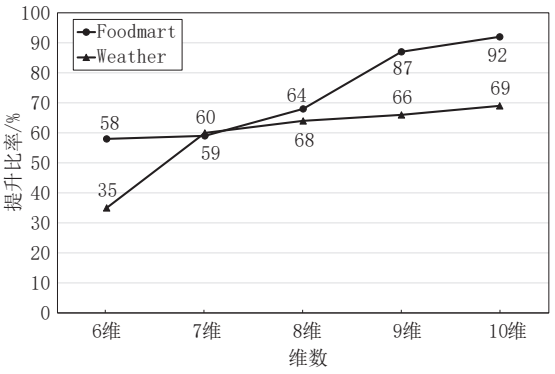


图17 Foodmart和Weather不同维度运行时间效率的提升

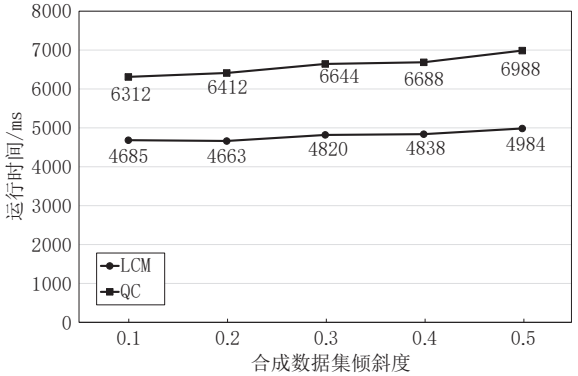


图18 合成数据集不同倾斜度下的运行时间

随着数据倾斜度增加，两种算法的运行时间都会增加，但是二者的增加速度都比较平稳．其中的原因是，随着数据变得倾斜，元组趋向于集中到数据空间更小的密集区域，使得该区域内等价类数量增加，但是，剩余的等价类分布在更稀疏的区域，导致在计算该区域时所消耗的时间增加．整体上来说，随着倾斜度的改变，LCM的效率还是优于QC．随着倾斜度的增加，LCM的运行效率相对于QC有小幅度提升．

7 结 论

数据立方体和频繁项集挖掘分别是数据仓库与数据挖掘领域中两种至关重要的技术．本文通过代数格的性质将二者结合起来进行系统性研究，提出了数据立方体和频繁项集计算的统一框架，给出了二者概念和计算间的映射关系，论证了二者的等价性．最后在真实数据集和合成数据集上进行实验，验证本文提出的映射的正确性和融合计算的有效性．首先，利用SQL Server分别对等列和不等列的事务集做冰山立方体计算，并将计算结果与LCM算法的频繁项集挖掘结果进行对比，发现对于相同事务集和支持度阈值，冰山立方体与频繁项集的计算结果完全一致，进一步证明了冰山立方体计算与频繁项集挖掘融合的正确性；其次，利用频繁项集挖掘的高效算法来实现数据立方体的计算，并且与经典的数据立方体计算方法进行对比，发现基于频繁项集挖掘算法的数据立方体计算在时间效率上要优于数据立方体的经典算法．

本文论证的数据立方体和频繁项集挖掘二者的等价性将较好地促进这两种重要技术的交叉融合．未来工作包括两个方面：(1)对更多的数据立方体与频繁项集挖掘计算方法建立统一关系；(2)利用本文所提出的统一性论证进行二者的相互促进，进一步提升数据立方体和频繁项集挖掘的计算效率，降低二者在计算时的时间开销和空间开销．

参 考 文 献

[1] Gray J, Chaudhuri S, Bosworth A, et al. Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data Mining & Knowledge Discovery, 1997, 1(1):29-53

[2] Han Jia-Wei, Kamber M, Pei Jian. Data mining: concepts and techniques. Third Edition. San Francisco: Morgan Kaufmann, 2011

[3] Ng R T, Wagner A, Yin Yu. Iceberg-cube computation with PC clusters//Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data. Santa Barbara, USA, 2001, 30(2): 25-36

[4] Wang Wei, Feng Jian-Lin, Lu Hong-Jun, et al. Condensed cube: an effective approach to reducing data cube size//Proceedings of the International Conference on Data Engineering. San Jose, USA, 2002:155-165

[5] Sismanis Y, Deligiannakis A, Roussopoulos N, et al. Dwarf: shrinking the PetaCube//Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. Madison, USA, 2002: 464-475

[6] Lakshmanan L V S, Pei Jian, Han Jia-Wei. Quotient cube: how to summarize the semantics of a data cube//Proceedings of the 28th International Conference on Very Large Databases. Hong Kong, China, 2002: 778-789

[7] Zou Shu-Zhi, Zhao Li, Hu Kong-Fa. A shell multi-dimensional hierarchical cubing approach for high-dimensional cube. Physics Procedia, 2012, 24:1715-1721

[8] Beyer K S, Raghu R. Bottom-up computation of sparse and iceberg cubes//Proceedings of the ACM SIGMOD International Conference on Management of Data. Philadelphia, USA, 1999, 28:359-370

[9] Lakshmanan L V S, Pei Jian, Zhao Yan. QC-trees: an efficient summary structure for semantic OLAP//Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. San Diego, USA, 2003: 64-75

[10] Agrawal R, Imieliński T, Swami A. Mining association rules between sets of items in large databases//Proceedings of the ACM SIGMOD International Conference on Management of Data. Washington, USA, 1993, 22(2):207-216

[11] Han Jia-Wei, Pei Jian, Yin Yi-Wen. Mining frequent patterns without candidate generation//Proceedings of the ACM SIGMOD International Conference on Management of Data. Dallas, USA, 2000, 29(2): 1-12

[12] Zaki M J. Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering. 2000, 12(3): 372-390

[13] Uno T, Asai T, Uchida Y, et al. LCM: an efficient algorithm for enumerating frequent closed item sets//Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations. Melbourne, USA, 2003, 90

[14] Uno T, Kiyomi M, Arimura H. LCM ver. 3: collaboration of array, bitmap and prefix tree for frequent itemset mining//Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations. Chicago, USA, 2005: 77-86

[15] Janostik R, Konecny J, Krajča P. LCM from FCA point of view: a CbO-style algorithm with speed-up features. International Journal of Approximate Reasoning. 2022, 142: 64-80

[16] Zaki M J, Hsiao C J. Efficient algorithms for mining closed itemsets and their lattice structure. IEEE Transactions on Knowledge and Data Engineering. 2005, 17(4): 462-478.

[17] Li Sheng-En, Wang Shan. Research on closed data cube technology. Journal of Software, 2004(08): 1165-1171 (in Chinese)
(李盛恩, 王珊. 封闭数据立方体技术研究. 软件学报, 2004(08): 1165-1171)

[18] Xiang Long-Gang, Gong Jian-Ya. A highly condensed and semantics-preserving data cube. Journal of Computer Research and Development, 2007(05): 837-844 (in Chinese)
(向隆刚, 龚健雅. 一种高度浓缩和语义保持的数据立方. 计算机研究与发展, 2007(05): 837-844)

[19] Shi Zhi-Bin, Huang Hou-Kuan. Reductive data cube based on formal concept analysis. Journal of Computer Research and Development, 2009, 46(11): 1956-1962 (in Chinese)
(师智斌, 黄厚宽. 基于形式概念分析的约简数据立方体研究. 计算机研究与发展, 2009, 46(11): 1956-1962)

[20] Garnaud E, Maabout S, Mosbah M. Functional dependencies are helpful for partial materialization of data cubes. Annals of Mathematics and Artificial Intelligence. 2015, 73:245-274

[21] Wang Quan-Kun, You Jin-Guo, Zou Ben-Yuan, et al. Reduced quotient cube: maximize query answering capacity in OLAP. IEEE Access, 2021, 9: 141524-141535

[22] Afrati F N, Sharma S, Ullman J R, et al. Computing marginals using MapReduce. Journal of Computer and System Sciences, 2018, 94: 98-117

[23] Lee S, Kang S, Kim J, et al. Scalable distributed data cube computation for large-scale multidimensional data analysis on a spark cluster. Cluster Computing, 2019, 22: 2063-2087

[24] Silva R R, Hirata C M, de Castro Lima J. Big high-dimension data cube designs for hybrid memory systems. Knowledge and Information Systems, 2020, 62: 4717-4746

[25] Xie Xi-Ke, Zou Kai, Hao Xing-Jun, et al. OLAP over probabilistic data cubes II: parallel materialization and extended aggregates. IEEE Transactions on Knowledge and Data Engineering, 2020, 32(10):1966-1981

[26] Zhang Yi-Qun, Ordonez C, García-García J, et al. The percentage cube. Information Systems, 2019, 79: 20-31

[27] Liu Jun-Yu, Jia Xiu-Yi. Multi-label classification algorithm based on association rule mining. Journal of Software. 2017, 28(11):2865-2878 (in Chinese)
(刘军煜, 贾修一. 一种利用关联规则挖掘的多标记分类算法. 软件学报. 2017, 28(11):2865-2878)

[28] Agrawal R, Srikant R. Fast algorithms for mining association rules//Proceedings of the 20th International Conference on Very Large Data Bases. Santiago, Chile, 1994: 487-499

[29] Yu Zi-Qiang, Yu Xiao-Hui, Dong Ji-Wen, Wanag Lin. Distributed mining of frequent co-occurrence patterns across multiple data streams. Journal of Software. 2019, 30(4): 1078-1093 (in Chinese)
(于自强, 禹晓辉, 董吉文, 王琳. 分布式多数据流频繁伴随模式挖掘. 软件学报. 2019, 30(4): 1078-1093)

[30] Zhang Shao-Xue, Wang Li-Zhen, Chen Wen-He. CPM-MCHM: a spatial co-location pattern mining algorithm based on maximal clique and hash map. Chinese Journal of Computers. 2022, 45(3): 526-541 (in Chinese)
(张绍雪, 王丽珍, 陈文和. CPM-MCHM:一种基于极大团和哈希表的空间并置模式挖掘算法. 计算机学报. 2022, 45(3): 526-541)

[31] Liang Wen-Juan, Chen Hong, Zhao Su-Yun, Li Cui-Ping. A differentially private scheme for top-k frequent itemsets mining over data streams. Chinese Journal of Computers. 2021, 44(4): 741-760 (in Chinese)
(梁文娟, 陈红, 赵素云, 李翠平. 一种面向数据流top-k频繁模式发布的差分隐私保护方案. 计算机学报. 2021, 44(4): 741-760)

[32] Ouyang Jia, Yin Jian, et al. Transaction data collection for itemset mining under local differential privacy. Journal of Software. 2020,32(11):3541-3562 (in Chinese)
(欧阳佳, 印鉴等. 面向频繁项集挖掘的本地差分隐私事务数据收集方法. 软件学报. 2020, 32(11): 3541-3562)

[33] Zhang Jing-Tian, Wu Sai, et al. Unexpected subgroup mining in multi-dimensional dataset. Chinese Journal of Computers. 2019, 42(8): 1671-1685 (in Chinese)
(张静恬, 伍赛等. 基于多维数据集的异常子群发现技术. 计算机学报. 2019, 42(8): 1671-1685)

[34] Li Ling, Yin Ying, Zhao Yu-Hai, Wang Guo-Ren, Dong Xiang-Jun. An efficient distributed algorithm for large-scale graph data mining based on decoupled summary subgraph. Chinese Journal of Computers. 2020, 43(7): 1183-1198 (in Chinese)
(李玲, 印莹, 赵宇海, 王国仁, 董祥军. 基于解耦概要图的大规模图数据高效分布式挖掘算法. 计算机学报. 2020, 43(7): 1183-1198)

[35] Cheng Gong, Liu Da-Xin, Qu Yu-Zhong, et al. Fast algorithms for semantic association search and pattern mining. IEEE Transactions on Knowledge and Data Engineering. 2021, 33(4):1490-1502

[36] Tang Xiao-Chun, Fan Xue-Feng, Zhou Jia-Wen, Li Zhan-Huai. An algorithm based on dataflow model for mining frequent patterns from a large graph. Chinese Journal of Computers. 2020, 43(7): 1293-1311 (in Chinese)
(汤小春, 樊雪枫, 周佳文, 李战怀. 基于数据流的大图中频繁模式挖掘算法研究. 计算机学报. 2020, 43(7): 1293-1311)

[37] Qu Wen-Wen, Yan Da, Guo Gui-Mu, et al. Parallel mining of frequent subtree patterns//Proceedings of the Software Foundations for Data Interoperability and Large Scale Graph Data Analytics. Tokyo, Japan, 2020, 1281: 18-32

[38] Messaoud R B, Rabaséda S L, Boussaid O, et al. Enhanced mining of association rules from data cubes//Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP. Arlington, USA, 2006:11-18

[39] Hu Xiao-Chun. Research of online analytical mining technology based on data warehouse [M. S. dissertation]. Xi' an Technological University, Xi'an, 2012 (in Chinese)
(胡小春. 基于数据仓库的联机分析挖掘技术的研究[硕士学位论文]. 西安工业大学, 西安, 2012)

[40] Khadija L, Omar E B, Mohammed R. OLAP cube partitioning based on association rules method. Applied Intelligence. 2019, 49(2): 420-434

[41] Han Jia-Wei, Pei Jian, Dong Guo-Zhu, et al. Efficient computation of iceberg cubes with complex measures// Proceedings of the ACM SIGMOD International Conference on Management of Data. Santa Barbara, USA, 2001, 30(2): 1-12

[42] Xin Dong, Han Jia-Wei, Li Xiao-Lei, et al. Computing iceberg cubes by top-down and bottom-up integration: the StarCubing approach. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(1): 111-126

[43] Shao Zheng, Han Jia-Wei, Xin Dong. MM-Cubing: computing iceberg cubes by factorizing the lattice space//Proceedings of the 16th International Conference on Scientific and Statistical Database Management. Santorini, Greece, 2004: 213-222

[44] Xin Dong, Shao Zheng, Han Jia-Wei, et al. C-Cubing: efficient computation of closed cubes by aggregation-based checking//Proceeding of the 22nd International Conference on Data Engineering. Atlanta, USA, 2006: 4-4

[45] Brin S, Motwani R, Ullman J D, et al. Dynamic itemset counting and implication rules for market basket data// Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data. Tucson, USA,1997, 26(2): 255-264

[46] Savasere A, Omiecinski E, Navathe S B. An efficient algorithm for mining association rules in large databases//Proceedings of the 21th International Conference on Very Large Data Bases. Zurich, Switzerland, 1995: 432-444

[47] Pasquier N, Bastide Y, Taouil R, et al. Discovering frequent closed itemsets for association rules//Proceedings of the 7th International Conference on Database Theory. Jerusalem, Israel, 1999, 1540: 398-416

[48] Pei Jian, Han Jia-Wei, Mao Run-Ying. CLOSET: an efficient algorithm for mining frequent closed itemsets//Proceedings of the 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. Dallas, USA, 2000, 4(2): 21-30

[49] Grahne G, Zhu Jian-Fei. Efficiently using prefix-trees in mining frequent itemsets//Proceedings of the IEEE ICDM 2003 Workshop on Frequent Itemset Mining Implementations. Melbourne, USA, 2003, 90:65-75

[50] Lucchese C, Orlando S, Perego R. DCI Closed: a fast and memory efficient algorithm to mine frequent closed itemsets// Proceedings of the IEEE ICDM 2004 Workshop on Frequent Itemset Mining Implementations. Brighton, UK, 2004

[51] Sarawagi S, Thomas S, Agrawal R. Integrating association rule mining with relational database systems: alternatives and implications. Data Mining and Knowledge Discovery, 2000, 4: 89-125

[52] Hahn C J, Warren S G, London A J. Edited synoptic cloud reports from ships and land stations over the globe, 1982-1991. Environmental Sciences Division. 1996, 4367: 1-43

XU Jing-Wen, M. S. candidate. Her main research areas include big data and data mining.

YOU Jin-Guo, Ph. D., professor. His main research areas include big data, cloud computing and data warehouse.

WANG Quan-Kun, M. S. candidate. His main research interests include big data and cloud computing.

HUANG Xing-Rui, M. S. candidate. His main research areas include data mining and machine learning.

JIA Lian-Yin, Ph. D., associate professor. His main research interests include database, data mining, information retrieval and parallel computing.

Background

A data cube is an important multi-dimensional data model in data warehouse and OLAP. How to speed up its computing efficiency and reduce storage cost is a crucial research issue in this field. Association mining is considered to be a key task in data mining, because of its capability to discover the relationship between different items from a large amount of data. Data cube computing and association mining have different tasks and goals. They analyze data from different angles and obtain different information contained in the data. However, data cubes and association mining construct similar algebraic lattice structures based on their data cells and itemsets respectively. Therefore, it is possible to combine the two and study them according to the properties of algebraic lattices.

Most related work only considers data cubes as a tool to assist association mining algorithms or introduce association mining to discover the association relationship of the dimensional data of data cubes. There is no analysis of the intrinsic relationship between the two and no comprehensive mapping of the two to achieve a unified study. By further studying the association mining process and the data cube computing methods, this paper focuses on the inherent connection of the data from two perspectives and establishes a unified calculation for the many computation properties and

methods of data cube computing and association mining, which can further clarify the data characteristics and improve the efficiency of the data cube aggregation operation and association mining. On the basis of these results, this paper further generalizes the relevant concepts. Specifically, the correlation between the classic computation methods of the data cubes such as Iceberg Cube, Condensed Cube, and Quotient Cube, and the association mining methods is established and the effectiveness of unified computing is further proved by a series of comprehensive experiments.

This work is supported by the National Natural Science Foundation of China (No. 62062046 and No. 61462050).