计算机科学 1998 Vol. 25№. 1

# KDD查询语言研究

KDD Query Language: A Perspective



冯玉才 冯剑琳

(华中理工大学计算机系 武汉 430074)

**摘 要** This paper mainly give a comprehensive perspective of KDD query language. We describe the KDD tasks and process, so the considerations in a design of KDD query language are shown. At the same time we also introduce a relation Data Mining Query Language, DMQL, it may serve as an interesting example for our discussion. Finally, we give many considerations of the combination of KDD with Deductive and Object-Oriented Databases.

**关键词** Knowledge discovery, Data mining, KDD query Language, Deductive databases, Object-oriented

## 1 引言

知识发现和数据开采(简称为KDD)就是从大量数据中析取事先隐藏未明却又潜在可用的模式(patterns)[6]。然而目前KDD领域还缺乏独立性,大多数从事机器学习(machine learning)领域的人把数据库只看作是大型文件。在他们看来,当所考虑的数据的量特别巨大时,机器学习问题就变成了数据库开采问题。结果许多人怀疑是否有必要因为数据集尺寸的大小而使KDD成为一个独立的研究领域,他们争辩道,机器学习处理或者应该处理KDD问题。因此T. Imielinski在[3]中呼吁从文件开采过渡到数据库开采,他认为今天的KDD系统靠现存的SQL支持与数据库交互作用正如六十年代的商业应用通过COBOL调用VSAM文件一样犯着同样的错误。

KDD之所以被认为具有令人兴奋的研究前景,是因为KDD能够获得广泛的应用,如决策支持、市场策略等等。面对汹涌而来的大量数据,对KDD应用的极大需求很有可能促使新一代数据库系统的形成。今天的数据库系统无论在性能效率方面还是在目前使用的API方面都不是为了支持KDD应用的,为KDD提供数据库支持就好比Codd把六十年代早期的VSAM文件系统带进到关系数据库系统的工作。也只有抓住这一巨大机遇取得突破性的进展,KDD才能不致于继续徘徊在机器学习和概率统计分析之间[3]。

商业应用的飞速发展促使人们如同当初Codd一样去寻找一些基本原语以使得应用开发更加快速和方便。Codd的努力导致了关系代数和关系演算等方法,并最终产生了SQL和关系API。关系语言核心较早地得到标准化,为关系系统的研究、开发和应用提供了一个坚实的基础,极大地促进了关系系统的商业化和广泛应用。目前的KDD研究和开发实践表明KDD需要覆盖各种各样不同的应用任务,从数据一般化到开采关联规则、数据分类,或者诸如进化规则、序列模式(sequential patterns)等等特定的模式。因此界定KDD所需的基本原语,设计一种通用全面的KDD查询语言将是极迫切而又极富挑战性的任务。

## 2 KDD的基本概念和任务

KDD的目的就是通过发现令人感兴趣的模式来帮助人们理解大量的原始数据。Frawley等人在[5]中对模式给出了如下定义:

给定一个事实(数据)集,一个语言L,以及一些对可信度C的量度,一个模式S就是L中的一个陈述,S以可信度C来描述F的一个子集Fs中的关系,并使得S要易于对Fs中所有事实的简单枚举。

对于KDD应用来说,事实集F通常都是特别巨大的,而发现的结果只在统计学的意义上是有效的,用户旨在寻找对有意义的相关数据部分成立的知识,而不一定要考虑所有的数据。因此所谓令人感兴趣的模式也就因人而异,各取所需。

由于数据量特别巨大，为避免如同人工智能中的组合爆炸问题，任何大型的商用 KDD 系统必然需要某种程度的用户参与。S. C. Yoon 等人提出 KDD 系统应该是一个自顶向下、交互式和增量式的发现系统[4,6]。"自顶向下"的意思是指我们对数据库提出特定的问题来计算和发现知识，提供信息来说明发现引擎应该搜索什么。"交互式"则是说一个知识分析员或用户重新检查输出后，他们就可以形成一个新的问题集以改良搜索或者至少可以更详尽地描述查找的某些方面。如果他们对发现的结果还不满意，他们可以调整输入信息以期发现更令人感兴趣的模式。这种知识分析员或用户与 KDD 系统之间的交互作用能够把搜索聚焦在令人感兴趣的模式中，这一点对于处理大量数据是至关重要的。"增量式"意为，当数据库被修改之后，我们可以很容易地修正以前导出的知识，因为我们只需要检查与数据库修改相关的那些知识。这种方法也使得我们易于利用旧知识来发现新知识。事实上，从数据库中发现的知识只反映了数据库的当前状态。为了维护已发现的知识的稳定性和可靠性，我们必须收集相当长的一段时间里的一大批数据，因此增量式更新技术将日益具有重要意义。

发现知识对大多数未来的 KDD 应用来说并不是最终目标，而是要将发现出来的知识结合到 KDD 应用中去并产生效益，实现某个目标。因此发现的结果应该针对目的不同的用户表示成适当的形式。通常发现的结果都是表示成某种形式的规则。规则的逻辑表示形式对于计算来说是很自然的，如基于一阶谓词逻辑，用于开采的数据和发现出来的规则可以以一种统一的框架来表示[2]。而采用某种"IF…THEN…"形式表述的规则，也是利于人们的理解的。但是人们还可能希望以更生动直观的形式以各种不同的视角来看待所发现的知识，甚至还要能看到原始数据是如何支持发现出来的知识的，因此灵活方便的图形用户界面 GUI 将是 KDD 应用必不可少的。

在 KDD 发现知识的过程中，其核心就是那些用来发现各种各样的模式或规则的算法。然而在获取有用的知识的整个问题当中，对模式或规则的推理仅仅还只是一小部分[7]。大致说来，KDD 过程包括以下任务：

1)数据收集和纯化。对将用于数据开采的原始数据进行某种预处理，说明 KDD 过程的有关数据以及必需的背景知识，还有附加说明或例外处理等等。纯化就是对原始数据进行"去噪"，如 H. Lu 在[8]中使用特征选择使得分类算法更加有效。

2)选择模式的发现方法。说明要发现哪一种知识以及有关参数的选择。如要发现关联规则，就要说明那些令人感兴趣的规则的最小支持度(minsup)和最小可信度(minconf)等有关阈值。

3)发现模式。调用相应的算法发现所需的知识，我们可以用数据开采这一术语来指 KDD 过程的这一模式析取部分[5]。

4)发现结果的后处理。把发现出来的结果表示成适当的形式，对真正令人感兴趣的模式进行选择等等。

5)将发现出来的知识加以利用。只有将所发现的知识与有关应用相结合，才能实现 KDD 的意义。如 S. C. Yoon 等人利用所发现的规则进行语义查询优化，而 K. Hätönen 等人则是利用所发现的知识来过滤冗余的报警信号[4,7]。

## 3 数据开采语言 DMQL

数据开采语言 DMQL 是 J. Han 等人为了开采关系数据库中各种不同的知识而设计的一种试验性语言，它的一部分已经在 DBMiner 系统中实现，用于交互式地开采多层次知识(multiple-level knowledge)[1]。我们将简要介绍 DMQL 的设计指导原则、语法以及概念层次和对 GUI 功能组成的考虑。我们可以将 DMQL 当作一个有趣的参照例子，以供进一步界定 KDD 查询语言基本原语的发现和探讨。

### 3.1 DMQL 的设计指导原则

1)应该在数据开采请求中说明用于数据开采的相关数据集。

2)应该在数据开采请求中说明想要开采的知识的种类。

3)数据开采过程中应该总是有可能运用相关的背景知识。

4)数据开采的结果应该能用较概括的或多层次概念的术语来表述。

5)应该能够说明各种各样的阈值，使得可以很灵活地过滤掉那些不是很令人感兴趣的知识。

### 3.2 DMQL 的语法

DMQL 采用了类似 SQL 的语法以适应在高级语言的水平上进行数据开采并与关系查询语言 SQL 保持自然的融合。

DMQL 是采用一种扩展的 BNF 语法来定义的，其中"【】"表示 0 或 1 次出现，"{}"表示 0 次或更

多的出现。用大写表示关键字，则 DMQL 可以如下表述：

```
〈DMQL〉::=
    USE DATABASE 〈database-name〉
    {USE HIERARCHY 〈hierarchy-name〉 FOR
        〈attribute〉}
    〈RULE-SPEC〉
    RELATED TO 〈attr-or-agg-list〉
    FROM 〈relation(s)〉
    [WHERE 〈condition〉]
    [ORDER BY 〈order-list〉]
    {WITH [〈kinds-of〉]THRESHOLD=〈thresh-
        value〉
    [FOR 〈attribute(s)〉]}
```

从字面上我们可以获得一个大致的了解，详情可参阅[1]。对于〈RULE-SPEC〉，DMQL 目前只考虑了下述几种规则的说明：数据一般化、特征规则(characteristic rules)、差别规则、分类规则和关联规则。

### 3.3 概念层次

概念层次用于表达控制一般化过程所必需的背景知识。依据从一般化到特殊的顺序，不同层次的概念经常组织成一个概念分类的格(lattice)。最一般化的概念就是内涵为空的描述(用保留关键字“ANY”表示)，而最特殊的概念就是数据库中特定的属性值。利用一个概念层次，我们就可以将所发现的规则表述成简洁明了、术语更一般化的形式。

{freshman, sophomore, junior, senior}⊂undergraduate
{M.S, M.A, Ph.D}⊂graduate
{undergraduate, graduate}⊂ANY(status)

图1 一个概念层次

图1所示为一个描述学生状态的概念层次表，其中 A⊂B 表示 B 是 A 的一个一般化。如果我们将概念层次以树的形式表示出来，就形成了一个概念树。如下即为学生状态的一个概念树。

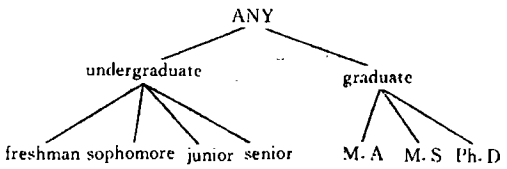


图2 一个学生状态的概念树

### 3.4 GUI 的功能组成

J. Han 等人根据他们的研究和开发经验，认为数据开采 GUI 大致应该包括下述功能组成：

1)数据收集，也就是收集相关数据集的界面。这样的一个界面很象一个构造关系查询的 GUI。

2)数据开采结果的表示，这是一个以不同形式显示数据开采结果的界面，包括表格、图形、曲线、视象等等。

3)数据开采原语的操纵，这包括对不同阈值的动态调整，概念层次的选择、显示和修改等等。

4)交互式多层次开采，即使得被发现的知识能够在不同层次上以不同的视角提供给用户。

5)其它各种辅助信息，这包括联机手册、帮助、调试和其它交互式图形工具。

## 4 KDD 与演绎和面向对象数据库的结合

演绎数据库就是在数据库中引入逻辑推理能力，面向对象数据库则通过面向对象的概念强有力地支持各种复杂应用领域中的数据建模要求，下面我们将讨论演绎和面向对象数据库框架中的 KDD。

### 4.1 KDD 与演绎数据库

演绎数据库的逻辑数据语言具有说明性的语义，而且基于规则的表达方式简洁明了，易于被人们理解；同时它又具有很强的语义表达能力，而传统的 SQL 是不能表达递归查询的，引入演绎推理机制，可以增加 SQL 的表达和计算能力，从而克服关系数据库中的阻抗不匹配问题。这些都是可以在设计 KDD 查询语言时给人以启发的。

在 J. Han 等人提出的 DMQL 中，RULE-SPEC 是以保留关键字来说明用户希望开采的知识的种类，如数据一般化和特征规则等，这种静态的方式缺乏一定的灵活性。事实上可以引入逻辑规则来说明用户想要发现的模式，这就产生了具有更高抽象层次的所谓“模式模板”(pattern templates)的概念[6]，我们希望它能表达一般化令人感兴趣的模式。例如我们考虑如下一个模板：

$$P(X,Y) \wedge Q(Y,Z) \Rightarrow R(X,Z)$$

其中 P,Q,R 是谓词。按照演绎数据库的观点，这是一条表示传递闭包的规则。这样的一个模板就定义了一个规则集，所有形如

$$p(X,Y) \wedge q(Y,Z) \Rightarrow r(X,Z)$$

的规则都可以看作是该模板的一个实例。因此我们应该考虑在 KDD 查询语言中引入表达逻辑谓词的机制。

现在我们来考虑演绎数据库中的 KDD，这包括两种情形：①在演绎规则所定义的数据上开采知识规则。②在已有的规则上开采新的规则。在第一种情形中，首先要执行一个演绎查询生成相应的 IDB 数据库，也就是与 KDD 任务相关的数据集，然后即可对这些数据进行开采了。在第二种情形中，可以考虑把 KDD 已经发现出来的结果加入到演绎规则集中去，然后对所有的规则进行开采以发现新的规则。如果有必要的话，显然可以重复上述过程直到发现用

户想要的规则。这一点可能是KDD与演绎数据库相结合具有潜在意义的地方。KDD为演绎推理发现新的可用的规则，由于KDD发现的结果具有概率统计的意义，所以可能需要引入某种确认机制，挑选出真正可用的规则来。这一点可以考虑借助用户或系统自动的反馈来实现。另一方面，演绎推理是否可以结合到KDD的开采算法中去以提高KDD开采的效率呢？一种可能就是利用演绎机制导出KDD开采任务所需的相关数据集，显然它这时就是一个IDB数据库。

### 4.2 KDD与面向对象数据库

相对于传统数据库来说，面向对象数据库具有更强的语义建模能力，能表达复杂对象之间的复杂联系。这主要得益于它所使用的面向对象的概念，如对象标识、封装、继承、多态性等等。在面向对象数据库中具有相同的实例变量和方法的许多对象组成一个类，类与子类之间的关联构成了类层次。子类继承超类的所有属性，即它的结构(型)和行为(方法)，同时子类也可有自己的属性。

如前所述，KDD的核心就是开采各种各样规则或模式的算法。这些发现算法可以分为两类。其中一种可以称为基于概念一般化的发现，它依赖于对概念(属性值)的一般化，从而在一个较高的概念层次上归纳出隐藏在大量数据中的某种规律性的知识，如DBLean系统就是基于面向属性的方法进行知识发现的[2]；另一种则不需要概念的一般化，它直接在同一个概念层次上发现某种具有较强规律性的规则，如发现关联规则[9]。然而这种在同一基本概念层次上的开采过于简单，很有可能发现不了一些规则或者发现一些无意义的、冗余的规则。正是有鉴于此，R. Srikant等人在[9]中引入了一般化关联规则(generalized association rules)的开采问题，即给定一个大规模交易(transaction)数据库，每一个交易包括一些相关的项目，在这些项目上有一个is-a概念层次，需要我们发现is-a概念层次的任意层次上的项目之间的关联规则。事实上较低层次上的规则可能不能获得最小支持，因而许多有意义的规则可能就被埋没了；而通过is-a概念层次得出的规则具有更高的概括力，能更深刻地揭示项目之间的内在联系，因而这些规则也就更具有适应性，同时我们也就可以过滤掉一些无意义的规则或者较低层次上冗余的规则。因此描述相关背景知识的概念层次对于KDD来说将是极其重要的和必不可少的。我们讨论KDD与面向对象数据库的结合，就是要指出聚集了面向对象概念强大语义能力的类层次将对描述背景知识的概念层次提供极其自然的支持，概念层次本身将成为一个类层次或者可以从描述面向对象基本模式的类层次中导出。例如，图2中的概念树就可以定义成为一个类层次，这样就不再需要用A⊂B表示B是A的一个一般化来构造概念层次了。

在面向对象的框架中，我们可以把用于数据开采的原始数据和相应的开采算法封装在一起，构成一个类，并通过继承机制构成类层次。如果有必要的话，我们可以在子类中定义新的开采算法。利用多态性的概念，各种各样的对象激活各自对应的开采算法，使得各种各样的KDD应用对用户而言是透明的，从而不再需要如DMQL中RULE-SPEC一样显式地说明要开采的规则的种类。因此采用面向对象概念的KDD查询语言也将具有一般面向对象语言所有的诸如模块性、可重用性等优点。

**结束语** 本文对KDD查询语言作出了一些分析和探讨。事实上演绎和面向对象概念的结合将提供更强的语义和计算能力。例如，J. Han等人在[1，2]中提出了一个迄今尚未解决的重要问题，即概念层次的自动发现。相对较强的背景知识不仅可以提高发现过程的效率，而且也更能体现用户的意向，从而可以高效地获得他们正想要的规则。如前所述，类层次可以用来更自然地支持概念层次，而通过定义一些演绎规则，就可以推理出用户站在不同的视角所提出需要的概念层次来。这样概念层次的发现本身也可以被当作是一个知识发现过程。因此在对象演绎数据库系统(DOODB)中的KDD查询语言将更具有挑战性，同时也将更具有发展潜力。

## 参考文献


[1] J. Han et al., DMQL: A Data Mining Query Language for Relational Databases, SIGMOD' 96 DMKD, Workshop on Research Issues on Data Mining and Knowledge Discovery

[2] J. Han et al., Knowledge Discovery in Databases: An Attribute-Oriented Approach, VLDB'92

[3] T. Imielinski, From File Mining to Database Mining, SIGMOD'96 DMKD

[4] S. C. Yoon et al., Mining Knowledge in Object-Oriented Frameworks For Semantic Query Optimization, SIGMOD'96 DMKD

[5] G. Piatetsky-Shapiro and W. J. Frawley, Knowledge Discovery in Databases, AAAI/MIT Press, 1991

[6] A. Silberschatz et al., User-Assisted Knowledge Discovery: How Much Should the User Be Involved, SIGMOD'96 DMKD

[7] K. Hatonen et al., Knowledge Discovery from Telecommunication Network Alarm Databases, IEEE'96 Data Engineering

[8] H. Lu et al., On Preprocessing Data for Effective Classification, SIGMOD'96 DMKD

[9] R. Srikant et al., Mining Generalized Association Rules, VLDB'95