结构化模式查询语言 SPQL 的实现

张玉芳　李亦彦
(重庆大学计算机学院　重庆 400044)

摘　要　数据挖掘系统中挖掘结果模式的存储迫切需要一个好的模式查询语言,并能对存储的挖掘结果较好地执行各种操作。本文给出了对存储的数据挖掘结果执行操作的结构化模式查询语言 SPQL 的实现方法。SPQL 是一个类自然语言同时遵循类 SQL 语法,SPQL 的具体实现通过存储过程来完成。利用 SPQL 语言可以方便地实现对三类数据挖掘结果模式(关联规则、分类和序列模式)的管理、查询和使用,从而使共享数据挖掘结果和提高数据挖掘效率成为可能。

关键词　数据挖掘,模式,存储过程

1　结构化模式查询语言 SPQL

结构化模式查询语言 SPQL(Structured Pattern Query Language)参照了数据挖掘语言 DMQL 描述挖掘过程的原则,借鉴 SQL 语言一体化、非过程化的思想,面向统一的三种数据挖掘结果模式的存储表示[6]。SPQL 采用类自然语言的形式对数据挖掘结果进行各种操作,具有类似 SQL 语言的语法。

模式存储系统采用三个关系表组成的模式库来统一存储关联规则、分类规则和序列模式,其关系模式表示为[6]:

PATTERN:
CONDITION (CLASS, ID, CONDITION, ATTRIBUTE, PARENT, NO)
ACTION (CLASS, ID, ACTION, ATTRIBUTE, PARENT)
PARA (CLASS, ID, CONF, SUP, TIME, INTV)

SPQL 提供了模式库结构定义、模式查询和模式更新三类操作,对模式的操作可以具体分解成对上述三个关系表的操作,而对每个关系表的操作,又可以用一系列关系运算来完成。首先将 SPQL 中的每一条语句分解成对三个关系表的操作,然后用一系列 SQL 语言按照一定的程序逻辑写出其实现过程。一条 SPQL 语句的实现,在 SQL 中被视为一个完整的事务;只有一系列 SQL 语句全部执行完毕,一条完整的 SPQL 语句才真正执行完毕。在关系数据库中,SPQL 的每一条语句的实现具体是利用存储过程来完成的。

2　模式库结构定义语句的实现

模式库结构定义的目的是为数据挖掘结果定义存储结构,包括 create_patternbase 语句和 drop_patternbase 语句。

2.1　create_patternbase 语句的实现

模式库存储结构的定义实际上在关系数据库中产生三个表,分别存储前提、行为和参数信息。在模式库结构定义语句中需要说明三个表的属性和类型,表名由模式存储系统根据模式库名自动生成,即系统根据<patternbase_name>来自动获取三个表的名字:<patternbase_name>_condition 为前提表名字,<patternbase_name>_action 为行为表名字,<patternbase_name>_para 为参数表名字。每个表中根据规则的分类由系统自动产生模式号属性 id、模式类型属性 class 以及属性 no。

在关系数据库中,create_patternbase 语句由三条 SQL 语句 create table 来实现,其存储过程描述如下:

```
Procedure Create_Patternbase(<create_patternbase >)
Begin
  Table_name(patternbase_name);
  CREATE TABLE : condition _table
    ( ID , no int, CLASS char, condition ,attribute, parent
  <type>
    {, <condi_name> <type>});
  CREATE TABLE : action_table
    ( ID int, CLASS char, action, attribute, parent <type
  >
    {, <act_name> <type>});
  CREATE TABLE :para_table
    ( ID int, CLASS char,
    <parameter_name> <type> {,<parameter_name>
  <type>});
End
Procedure table_name(patternbase_name);
Begin
  condition_table : = <patternbase_name>+"_ condition";//产生前提表名
  action_table : = <patternbase_name>+"_action";//产生行为表名
  para_table : = <patternbase_ name>+"_para";//产生参数表名
End
```

其中,用"<>"括起来的部分是从 SPQL 中获

得的参数。只有当三个关系表全部完成，所需要的模式库才真正建立成功。

2.2 drop_patternbase 语句的实现

模式库的删除操作实现非常简单，分别用三条SQL语句就可以完成，整个过程为：

```
Procedure Drop_Patternbase(<drop_patternbase >)
Begin
  Table_name(patternbase_name);
  DROP TABLE : condition_table;
  DROP TABLE : action_table;
  DROP TABLE : para_table;
End
```

3 select_pattern 语句的实现

模式查询语句 selectpattern 在模式库中查找符合给定条件的数据挖掘结果，它的实现同样涉及模式库中的三个表。其基本思想是：首先，系统根据模式查询语句中的<patternbase_name>提取要查找的三个关系表的名称；然后根据模式查询条件表达式<query_expression>分别在三个表中找出满足条件的模式号集合{ID}，返回的三个集合的交集就是要查找的模式号集合；最后根据选出的模式号集合返回符合条件的模式。整个过程描述如下：

```
Procedure Select_Pattern(<select_pattern>)
Begin
  Table_name(patternbase_name);
  class := <class_value> ;//确定模式的类型
    //在三个关系表中查找满足条件的模式号集合 result_
  id_set
  result_id_set :=
  find_id_set(<query_expression>, condition_table, ac-
  tion_table, para_table, class);
  result_patterns := gen_patterns(result_id_set);//根据
  result_id_set 产生模式
  return result_patterns;//返回模式集合
End
```

模式查询语句实现的关键和复杂之处是第二步 find_id_set 和第三步 gen_patterns()。

3.1 find_id_set( )的实现

find_id_set( )根据查询表达式<query_expression>中是否有 condition、action 和 parameters 来确定具体对哪几张表执行查询操作；根据查询表达式中的操作符及其值来产生查询结果；最后对查询结果执行交集运算。其过程描述如下：

```
Procedure Find_Id_Set (< query_expression >, condition,
action, para, class)
Begin
  result_id_set := {id | id 是模式库中的 id};
  If (<query_expression>中包含 condition 部分) Then
    {idset_of_condition: = find_in_condition (<operator
  >,<value_set>, condition, class);
  result_id_set := result_id_set∩idset_of_condition; }
  If (<query_expression>中包含 action 部分) Then
    {idset_of_action: = find_in_action ( <operator>,<
  value_set>, action, class);
  result_id_set: = result_id_set∩idset_of_action;}
  If (<query_expression>中包含 parameter 部分) Then
    {idset_of_para := find_in_para(<para_exp>, para,
  class);
  result_id_set: = result_id_set∩idset_of_para;}
  Return result_id_set;
End
```

结果集合 result_id_set 初始化为全体 id 集合，然后和每次的查询结果求交集获得最终的结果集合。Find 函数主要功能是在三个表中查找满足条件的 id 集合，其实现的基本思想是：在读取前提部分的查询表达式时将参加比较的参数集合视为一个关系 R(c)，其中属性 c 的类型和前提表 condition 属性类型相同。假设前提表的关系模式为 CONDITION(id, class, condition, attribute, parent, no)，行为表关系模式为 ACTION(id, class, action, attribute, parent)。具体实现步骤为：

第一步：对 CONDITION 关系执行投影操作，获得前提表中所有(id)构成的关系 ID(id)：ID (id) = ∏ID (CONDITION)。

第二步：用 ID 中的每一个 idi 作为条件在前提表 CONDITION (id, condition) 中做选择运算，CCONDITION (id, condition) = σCONDITION. ID=IDi (CONDITION)，每次选择产生一个新关系 CCONDITION(id, condition)，即选择出每条模式 ID 所对应的前提集合。

第三步：对每一条 ID 选择运算产生的结果关系 CCONDITION 和准备比较的参数关系 R 做自然连接运算：CONDITIONRESULT (condition) = CCONDITION >< R，产生结果关系 CONDITIONRESULT。

第四步：重复第二步，直至 ID 中所有元组操作完毕。最后获得选出的模式号集合。

交集运算方法描述如下：首先在 RESULT 表中存入全部 id 号，然后对每一部分的结果 id 集合表 IDRESULT 做如下操作：RESULT×IDRESULT，结果就是 RESULT 和 IDRESULT 的交集，然后从 RESULT 中删除不出现在交集中的元组，依此执行，最终的 RESULT 就是结果集合。

而对所获取的结果关系 CONDITIONRESULT(condition)根据操作符和值之间可能的匹配方式判断其中的 ID 是否满足条件，如果符合条件，则将其置于结果集合中，用于后面产生结果模式。可能存在以下四种匹配方式：

1)“完全匹配”的“与”

自然连接操作的结果将删除两个关系 CCONDITION 和 R 中 condition 不相等的元组；而判断自然连接运算是否删除元组，只要比较 CONDITIONRESULT 中的元组个数和 CCONDITION 中的元组个数是否相等，两者相等则表示这个 id 对应的前提集合和条件集合之间满足“完全匹配”的“与”关系。

2)“完全匹配”的“或”

如果 CCONDITION 的元组个数为 1 并且自然

连接运算后 CONDITIONRESULT 的元组个数也为 1，则表示此 ID 对应的前提和条件集合之间满足“完全匹配”的“或”关系。这种情况实际上也是针对单前提模式。

3)“部分匹配”的“与”

如果 CONDITIONRESULT 中的元组个数和 R 中的元组个数相等，则表示此 id 对应的前提和条件集合之间满足“部分匹配”的“与”关系，即前提集合中包含了条件集合，因此执行自然连接运算后，R 中的元组都应该保留下来。

4)“部分匹配”的“或”

只要 CONDITIONRESULT 中的元组个数大于或等于 1 即表示这个 id 对应的前提和条件集合之间满足“部分匹配”的“或”关系，即前提集合中至少包含条件集合中的一项，执行自然连接操作后所剩下的元组个数要大于或等于 1。

3.2 gen_patterns()的实现

结果集合是查询操作获得的以 id 表示的模式集合，gen_patterns 功能是根据(id)将模式产生便于用户理解的直观表示形式。显示的形式由用户通过 display in 〈result_form〉来指定，可以是表格方式或图形化方式。系统根据〈result_form〉中的指定部分建立 PATTERN_TABLE 表，然后根据 RESULT 表中的 ID 选择出需要显示的部分插入 PATTERN_TBALE 中。其中参数部分比较好实现，可以直接把参数选择出来填入 PATTERN_TABLE 的相应属性中，而前提部分和行为部分则要把多个项目拼接成 CONDITION 或 ACTION。

4 模式更新语句的实现

模式更新是对模式库中存储的数据挖掘结果执行日常维护操作，包括模式插入语句 insertpattern、模式删除语句 deletepattern 和模式修改语句 updatepattern。

4.1 insert_pattern 语句的实现

根据模式存储结构，插入一条模式实际上是分别插入模式的前提部分、行为部分和参数部分。基本思想是：首先根据模式的前提和行为条件在模式库中查找是否存在这条模式，如果本条模式已经存在，则仅向参数表插入参数；如果模式不存在，则分别往三个表插入。在插入过程中系统不仅要保证模式的唯一性即保证(id)不发生冲突，还要保证被插入模式的完整性，其中包括模式的前提部分和行为部分不能为空，模式的参数个数必须和模式库定义的参数数目相等。如果插入的模式少了某一项参数就不能被插入模式库，同时模式中各项目的值域必须和模式库定义的值域相同。插入工作完成后，系统将返回该模式的(id)，否则返回错误消息。其实现过程如下：

```
Procedure Insert_ Pattern(〈insertpattern〉)
Begin
  Table_ name(patternbase_ name);
  检查待插入模式的完整性，如果不满足则返回错误消息；
  If (patternbase 中不存在待插入模式) Then
  {
    SELECT max(ID) INTO :id FROM :condition_ table;
    id := id + 1;//生成 id
    For each premise item
      INSERT INTO: condition _ table
        (id, class, condition, attribute, parent, no, condi_
  name)
        VALUES(: id, class, condition, attribute, parent,
  no, condi_ name);
      For each action item
        INSERT INTO : action_ table(id, class, action, at-
  tribute, parent,
  act_ name)
        VALUES (: id, class, action , attribute, parent,
  act_ name);
    };
    For each para item
      INSERT INTO : para_ table(id, para) VALUES (:
  id, para);
End
```

4.2 delete_pattern 语句的实现

在模式库中查询有无待删除的模式，如果有则根据找到的(id)分别从三个关系表中删除具有这个(id)的记录即可。

4.3 update_pattern 语句的实现

模式更新语句可以看作是 delete_pattern 和 insert_pattern 两条语句的结合，只是在实现的过程中要记下(id)。具体实现方法略。

小结 本文针对关联规则、分类规则和序列模式的存储结构提出的结构化模式查询语言 SPQL，给出了 SPQL 语句在关系数据库中的具体实现方法。利用 SPQL 语言，可以实现对数据挖掘结果进行管理和查询，从而使共享数据挖掘结果和提高数据挖掘效率成为可能。

参 考 文 献

1 Han J, Fu Y, Koperski K, Wang W, Zaiane O. DMQL: A data mining query language for relational databases. In: 1996 SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'96), Montreal, Canada, June 1996

2 Imielinski T, Virmani A. MSQL: A query language for database mining, Data Mining and Knowledge Discovery, 1999, 3: 373～408

3 Meo R, Psaila G, Ceri S. A new SQL-like operator for mining association rules. In: Proc. 1996 Int. Conf. Very Large Data Bases, Bombay, India Sept. 1996. 122～123

4 Tsur D, Ullman D J, Abitboul S, Clifton C, Motwani R, Nestorov S. Query flocks: A generalization of association-rule mining, SIGMOD98, Seattle, Washington, June 1998

5 http://www.crisp-dm.org/

6 Xiong ZhongYang, Zhang Yufang, Cheng Daijie Cheng, Zhang Rui. The Storage Method of Data Mining Results. In: The Proceedings of ICMLC2003, China, 2003

7 Tang Rongjun, Xiong ZhongYang, Zhang Yufang. ARQL: An Association Rule Query Language for Association Rule Base. In: The Proceedings of DCABES2002, Wuxi, China, 2002