数据流查询语言的研究与实现

周 杰[1],毛宇光[1,2]
(1.南京航空航天大学 信息科学与技术学院,江苏 南京 210016;
2.南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093)

摘 要:分析了现有的数据流管理系统中的查询语言;举出了一个现实生活中数据流应用的例子;提出了数据流查询的应用场景。通过这个例子对数据流模型作了形式化的定义,并提出了如何通过窗口操作将流式数据转化为普通关系中的数据。最后提出了一种数据流查询语言 MYCQL(My Continue Query Language),并给出了 MYCQL 中相关的文法,并借助 Lex 和 Yacc 工具实现了从查询语言生成语法分析树。

关键词:数据流管理系统;窗口;MYCQL;文法
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2008)01-0014-03

Study and Implementation of Data Stream Query Language

ZHOU Jie[1], MAO Yu-guang[1,2]
(1.Coll. of Info. Sci. and Techn., Nanjing Univ. of Aeronautics and Astronautics, Nanjing 210016, China;
2.State Key Lab. for Novel Software Techn., Nanjing Univ., Nanjing 210093, China)

Abstract: In this paper, analyzed the query languages in the recent data stream management systems, then give an example of the application of the data stream. Via the example give a formal definition of the data stream model, and point out how to use window operator to translate the stream data into the traditional relation data. At last propose a language that supports data stream query. Call it My Continue Query Language(MYCQL), and give the regular language of the data stream query language, and use the tool of Lex and Yacc to generate grammar parse tree.

Key words: data stream management system; window; MYCQL; regular language

0 引 言

很多的应用是需要对大量的、流动的、变化的数据进行处理,比如:股市行情、对车流进行分析、网络上传输的媒体流、传感器网络上的传感器传输的数据等。传统的数据库管理系统是对存储在硬盘等外存介质上的静态数据进行查询等操作,它的这个特性使得它无法胜任对数据流的查询操作。传统数据库处理持久存储的数据处理模式已不能适应现在大量的实时在线应用的需求:数据到达是连续的,数据的查询处理要求是单步处理,既来不及储存也无法存储流数据。传统的结构化查询语言(SQL)无法用于对数据流的查询操作[1]。文中主要以 SQL 语言为基础,研究如何修改 SQL 语言,使之适应数据流的查询操作,并且提出了一种数据流的查询语言 MYCQL,最后说明了如何在数据流管理系统(DSMS)中实现这个查询语言,也就是从数据流查询语言生成语法分析树。

收稿日期:2007-03-20
基金项目:国家重点基础研究规划项目(G1999032701)
作者简介:周 杰(1983-),男,江苏吴江人,硕士研究生,研究方向为数据流;毛宇光,副教授,博士后,研究方向为数据库理论、特种数据库及多值逻辑。

1 相关工作

连续查询的概念是在 Tapestry[2]中首先被明确提出的,在这个系统中提出了一种基于 SQL 的连续查询语言 TQL。TQL 查询在每个时间片执行一次(这个时间片可以很小),就像 SQL 语言在数据库在那个时间片的快照的一次查询。最后将各个时间片的查询结果作集合的并集操作,这样来完成对数据流的查询操作。TelegraphCQ 系统[3]提出了基于窗口机制的连续查询语言。Gigascope 系统是针对网络监视应用设计的,在这个系统中提出一种类似 SQL 的语言 GSQL。Aurora 系统[4]提供了大量的操作符,从基本的数据流过滤操作到窗口操作和聚集操作,用户只需选择合适的操作符。Aurora 系统根据操作符来完成数据流的查询操作。STREAM 系统[5]提出了基于 SQL-99 的 CQL 语

言[6,7],支持从数据流到关系的窗口操作和从关系到数据流的操作。文中结合以上几个系统的优点,提出了一种通用的数据流查询语言。

2 MYCQL 连续查询语言

2.1 应用场景

此处举一个股市分析的例子,在现实生活中股市行情瞬息万变,在每个时间点价格经常在变化,可以将这样的数据看成是流式的数据。假如要查询在最近一小时内股票名为"XX"的价格大于股票名为"YY"的价格,并且"XX"的股票价格大于 10 元时的"XX"和"YY"股票的详细情况,这时候就可以使用类似于 SQL 的 MYCQL 连续查询语言。下面的讨论将对这个查询形式化。

2.2 数据流定义

就像传统的关系模型,每一个数据流都有一个包含了一组属性的固定模式,为了描述数据流单元的到达,假设有一个离散的、有序的时间域 T。一个时刻就是一个 T 中的值。为了简单时间域的表述,假设时间域是一组非负的整型数据{0,1,2,3,…}。0 表示初始时刻。时间域 T 是一个逻辑的时间,可以和现实世界中的物理时间不同。时间域 T 需要满足两个条件:

(1) 时间域 T 中的时间点是离散的;

(2) 时间域 T 中的时间点是有序的。

定义 1　数据流:一个数据流 S 是具有形式 $<s, t>$ 的数据的一个集合,其中 s 是符合数据流模式规定的元组,和传统关系型数据库中的元组是一致的。其中 $t \in T$ 是一个时间戳,表示元组 s 到达的时间。

一个数据流的单元 $<s,t> \in S$ 表示元组 s 在 t 时刻到达数据流 S,时间 t 不是数据流模式的一部分。

上述股市的数据流可以定义为:Stock(id int, name char(20), float price)。

2.3 将数据流映射到关系

将数据流映射到关系最常用的办法就是对数据流进行加窗口操作,如图 1 所示。

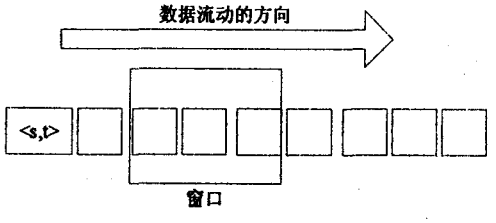

图 1　数据流的窗口操作

图中小的矩形框表示数据流 S 中的各个数据单元,每个数据单元用 $<s,t>$ 表示。通过窗口操作将所要查询的数据限定在窗口的范围内,这样就可以将潜在无限的数据化为有限的数据。这个窗口可以是基于时间的,也可以是基于元组的。因此在数据流查询语言中需要提供对这两种窗口的支持。

定义 2　基于时间的滑动窗口:一个在数据流 S 上的基于时间的滑动窗口,在连续查询语言中表示成 S[Time T]。其中 T 表示时间间隔。S[Time T] 的输出关系 R 为:

$R(t) = \{ s \mid <s, t'> \in S \wedge (t' \leqslant t) \wedge (t' \geqslant \max\{t - T, 0\})\}$。其中 $R(t)$ 表示在时刻 t 数据流 S 生成的关系集合。

下面有两种特殊的情况:

(1) 当 $T = 0$ 时,$R(t) = \{s \mid <s, t'> \in S \wedge (t' \leqslant t) \wedge (t' \geqslant \max\{t, 0\})\}$,

即:$R(t) = \{ s \mid <s, t'> \in S \wedge (t' \leqslant t) \wedge (t' \geqslant t)\}$

$R(t) = \{ s \mid <s, t'> \in S \wedge (t' = t)\}$

因此当 $T = 0$ 时,S[Time T] 输出的关系 R 为:在任一时刻 t,具有时间戳为 t 的数据流中的元组集合。

(2) 当 $T = +\infty$ 时,$R(t) = \{ s \mid <s,t'> \in S \wedge (t' \leqslant t) \wedge (t' \geqslant \max\{t - (+\infty), 0\})\}$,因为 t 为非负值,所以 $R(t) = \{ s \mid <s, t'> \in S \wedge (t' \leqslant t) \wedge (t' \geqslant 0)\}$。因此当 $T = +\infty$ 时,S[Time T] 输出的关系 R 为:在任一时刻 t,具有时间戳 $t' <= t$ 的数据流的元组的集合。

定义 3　基于元组的滑动窗口:一个在数据流 S 上的基于元组的滑动窗口,在连续查询语言中表示成 S[Tuple N],其中 N 为正整数。S[Tuple N] 的输出关系 R 为:$R(t) = \{ s_1, s_2, s_3, \cdots, s_n \mid <s_1, t_1> \in S \wedge <s_2, t_2> \in S \wedge <s_3, t_3> \in S \wedge \cdots \wedge <s_n, t_n> \in S, (t_1 \leqslant t_2 \leqslant t_3 \leqslant \cdots \leqslant t_n) \wedge t_n \leqslant t\}$。

当 $N = +\infty$ 时,根据上面的定义 S[Tuple N] 输出的关系为:在任一时刻 t,具有时间戳 $t' <= t$ 的数据流的元组的集合。因此当 $N = +\infty$, $T = +\infty$ 时,S[Tuple N] 等价于 S[Time T]。

上面的数据流要查询最近一个小时的股市,可以通过增加一个基于时间的滑动窗口来实现,如:Stock [Time 1hour]。

2.4 数据流的连接操作

在传统的 SQL 语言中支持表的连接操作,在数据流查询语言中同样也需要支持数据流的连接操作。假设有数据流 R 和 S,那么 R[Time T] $\triangleright\triangleleft_{A\theta B}$ S[Time T'] $= \{ (r,s) \mid r \in R(t) \wedge s \in S(t') \wedge r[A] \theta s[B]\}$,其中 $R(t)$ 和 $S(t')$ 是 R[Time T] 和 S[Time T'] 的输出关系。

因此上述的数据流查询可以用 MYCQL 表示为:

```
Select *
from Stock[Time 1hour]as StockA, Stock[Time 1hour] as StockB
where StockA. price > StockB. price and StockA. price > 10 and
StockA. name = 'XX'and StockB. name = 'YY';
```

3 MYCQL 语言的实现

3.1 Lex 和 Yacc 的工作原理

在原型系统中采用了 Lex 和 Yacc 来实现从查询语言到语法分析树的建立。Lex 和 Yacc 的工作原理如图 2 所示。

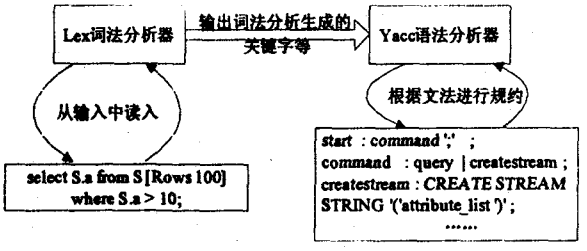

图 2 Lex 和 Yacc 工作的原理

其中 Lex 是词法分析器，负责从输入的查询语言中提取关键字和需要被识别的字符，并将提取到的字符等传递给 Yacc 语法分析器。Yacc 语法分析器则是根据文法规则进行规约。Yacc 通过寻找可以匹配目前为止所看到的标记的规则来工作。Yacc 处理语法分析程序时创建了一组状态，当语法分析程序读取标记时，每次它读取一个没完成规则的标记，就把它压入内部堆栈中并切换到一种反映它刚刚读取的标记的新状态。这个动作称为移进。当它发现组成某条规则的全部符号时，它就把右侧符号弹出堆栈，这个动作称为规约。Lex 和 Yacc 就是这样配合起来对输入的文法进行识别的。

3.2 规约文法

文中的文法支持数据流的连接操作、聚集操作、窗口操作等。在 Yacc 中使用的文法如下：

(1)SELECT 查询。

select_query → select_clause from_clause opt_where_clause opt_group_by_clause

select_clause → SELECT DISTINCT project_list | SELECT project_list | SELECT DISTINCT '*' | SELECT '*'

from_clause → FROM stream_list

opt_where_clause → WHERE condition_list | nothing

opt_group_by_clause → GROUP BY attributeibute_list | nothing

project_list → project_item ',' project_list | project_item

project_item → arith_expr | aggr_expr

stream_list → relation_variable ',' stream_list | relation_variable

attribute_list → attribute ',' attribute_list | attribute

attribute → STRING '.' STRING | STRING

nothing → /* epsilon 空 */

(2)条件表达式。

condition_list → condition AND condition_list | condition

condition → arith_expr LT arith_expr | arith_expr LE arith_expr | arith_expr GT arith_expr | arith_expr GE arith_expr | arith_expr EQ arith_expr | arith_expr NE arith_expr

(3)聚集表达式。

aggr_expr → COUNT'(' attribute')' | COUNT'(' '*' ')' | SUM '('attribute')' | AVG '('attribute')' | MAX '('attribute')' | MIN '('attribute')'

(4)窗口操作。

relation_variable → STRING '[' window_type']' | STRING '[' window_type ']' AS STRING | STRING | STRING AS STRING

window_type → TIME time | TUPLE INT | TIME UNBOUNDED

time → INT | INT SECOND | INT MINUTE | INT HOUR | INT DAY

(5)算术表达式。

arith_expr → attribute | const_value | arith_expr'+' arith_expr | arith_expr '-' arith_expr | arith_expr'*'arith_expr | arith_expr'/'arith_expr | '('arith_expr')'

const_value → QUOTE-STRING | INT | REAL

其中大写的表示规约的最终状态，而小写的表示规约的非最终状态。

在 Yacc 中每规约一个正则表达式都可以定义一个用户自定义的操作，就是在这个自定义操作中建立了语法分析树。

4 结束语

提出了一种新的数据流查询语言，并且借助 Lex 和 Yacc 实现了这种查询语言，实现了从用户输入到语法分析，生成语法分析树的过程。实现数据流查询语言是实现整个数据流管理系统的第一步，后面还需要对生成的语法树进行语义的分析，因为语法正确并不能保证语义是正确的。还需要生成查询计划，进行查询优化等。

参考文献：

[1] Golab L, Ozsu M T. Issues in data stream management[J]. SIGMOD Record, 2003, 32(2): 5 - 14.

[2] Terry D, Goldberg D, Nichols D, et al. Continuous queries over append - only databases[C]//In Proc. of the 1992 ACM SIGMOD Intl. Conf. on Management of Data. San Diego, California: [s.n.], 1992: 321 - 330.

[3] Chandrasekharan S, Cooper O, Deshpande A, et al. Tele-

(下转第 21 页)

$c\sqrt{N}d_q d$。

MREIB－DD在时间 T 内总通信量为:

$$\lceil CO_{\text{MREIB-DD}}\rceil \approx (c\sqrt{N}+1)D_{\text{dtat-an}}d_a + \frac{T}{a}d_t + Pc\sqrt{N}(d_p+d_r) + V(c\sqrt{N}d_n + Nd_b) + Q(d_m + d_x + 2c\sqrt{N}d_l) + c\sqrt{N}d_q d \quad (1)$$

3.2.2　GHT 和 MREIB－DD 最大总通信开销比较

GHT 算法的最大总通信开销(CO)为:

$$\lceil CO_{\text{GHT}}\rceil = (c\sqrt{N})D_{\text{total}}d + fMR^2 b\frac{T}{T_h} + c\sqrt{N}Qd_l + c\sqrt{N}d_q d \quad (2)$$

$$\lceil CO_{\text{GHT}}\rceil - \lceil CO_{\text{MREIB-DD}}\rceil = (c\sqrt{N})D_{\text{total}}d + fMR^2 b\frac{T}{T_h} - (c\sqrt{N})D_{\text{data-an}}d_a - Pc\sqrt{N}(d_p+d_r) - V(c\sqrt{N}d_n + Nd_b) - Q(d_m + d_x + c\sqrt{N}d_l) \quad (3)$$

所以 MREIB－DD 的总通信量小于 GHT。

4　总结及下一步工作

以无线传感器网络中数据分发算法的研究与改进为出发点,研究如何解决无线传感器网络中数据分发的能源有效性和负载均衡性的问题,在分析了无线传感器网络工作特点和现有数据分发算法的基础上,提出了基于 ILS 的最大剩余能量索引数据分发算法。对其性能进行了理论分析,证明其性能优于已有的 GHT 算法,能降低系统的能量消耗,延长系统的生存时间。下一步的工作主要是根据无线通信的理论(能量)模型,实际分析这两种算法,在特定应用背景下(例如楼宇监控)采集各种数据信息,计算其能量消耗的具体值,与 GHT 算法进行对比。并将此理论应用到实际的传感器节点上(例如 MicaZ)。

参考文献:

[1] Akkaya K, Younis M. A Survey on Routing Protocols for Wireless Sensor Networks[J]. in the Elsevier Ad Hoc Network Journal,2003(8):325－349.

[2] Akyildiz I,Su W,Sankarasubramaniam Y,et al. Wireless Sensor Networks: A Survey[J]. Computer Networks, 2002, 38(4):393－422.

[3] 孙利民,李建中,陈　渝,等.无线传感器网络[M].北京:清华大学出版社,2005.

[4] Ratnasamy S,Karp B,Yin L,et al. GHT:A Geographic Hash Table for Data－Centric Storage[C]//ACM International Workshop on Wireless Sensor Networks and Applications.[s.l.]:[s.n.],2002.

[5] Ghose A,Grobklags J,Chuang J. Resilient data－centric storage in wireless ad－hoc sensor networks[C]//Proceedings the 4th International Conference on Mobile Data Management (MDM'03).[s.l.]:[s.n.],2003:45－62.

[6] Greenstein B,Estrin D,Govindan R,et al. DIFS:A Distributed Index for Features in Sensor Networks[C]//First IEEE Ineternational Workshop on Sensor Network Protocols and Applications.[s.l.]:[s.n.],2003.

[7] Heinzelman W, Chandrakasan A, Balakrishnan H. Energy－Efficient Communication Protocal for Wireless Microsensor network[C]//Proc. of the Hawaii International Conference on System Sciences.[s.l.]:[s.n.],2000.

[8] Intanagonwiwat C,Govindan R,Estrin D. Directed Diffusion: A Scalable and Robust Communication[C]//IEEE Mobi－COM '00.[s.l.]:[s.n.],2000.

[9] Ye F,Luo H,Cheng J,et al. A Two－Tier Data Dissemination Model for Large－scale Wireless Sensor Networks[C]//ACM International Conference on Mobile Computing and Networking (MOBICOM'02).[s.l.]:[s.n.],2002:148－159.

[10] Intanagonwiwat C,Govindan R,Estrin D. Directed diffusion:A scalable and robust communication paradigm for sensor networks[C]//In:Pickholtz R,ed. Proc. of the 6th Annual Int'l Conf. on Mobile Computing and Networking. New York: ACM Press,2000:56－67.

[11] US Naval Observatory (USNO). GPS Operations[EB/OL]. 2001－04. http://tycho.usno.navy.mil/gps.html.

[12] Chong Chee－Yee,Kumar S P. Sensor Networks:Evolution, Opportunities,and Challenges[J]. Proceedings of the IEEE, 2003,91(8):1247－1256.

[13] Finn G G. Routing and addressing problems in large metropolitan－scale internetworks[R].[s.l.]: Information Sciences Institute,1987.

(上接第 16 页)

graphCQ: Continuous data flow processing for an uncertain world[C]//In Proc. of the 1st Conf. on Innovative Data Systems Research. Asilomar,CA,USA:[s.n.],2003:269－280.

[4] Carney D, Centintemel U, Cherniack M, et al. Monitoring streams－a new class of data management applications[C]// In Proc. of the 28th Int'l. Conf. on Very Large Data Bases. Hong Kong:[s.n.],2002:215－225.

[5] Arasu A, Babcock B, Babu S, et al. STREAM: The Stanford Data Stream Management System[EB/OL]. 2004. http://dbpubs.stanford.edu/pub/2004－20.

[6] Arasu A, Babu S, Widom J. CQL: A language for continuous queries over streams and realations[C]//In 9th Int'l. Workshop on Database Programming Languages. Potsdam, Germany:[s.n.],2003:1－11.

[7] Arasu A, Babu S, Widom J. The CQL Continuous Query Language: Semantic Foundations and Query Execution[R/OL]. 2003－10. Stanford University. http://dbpubs.stanford.edu/pub/2003－67.