

数字图书馆 XMARC 文档的关系模式与数据获取研究

王兰成

(解放军南京政治学院上海分院 上海200433)

摘 要 本文根据 CNMARC 元数据设计了基于 XMARC 资源描述框架的 XML DTD, 具体研究了 XMARC 文档的关系存储及其实现算法, 最后通过 MARC 的数据转换实现了 XMARC 文本数据的自动转换。

关键词 数字图书馆, XMARC 元数据, 关系模式

Study to the Relation Schema and Data Excavator on XMARC Document of Digital Library

WANG Lan-Cheng

(Department of Information Mangement, Nanjing Political Institute Shanghai Branch, Shanghai 200433)

Abstract This paper has designed the solving schema of XML DTD that describes frame based on XMARC resource according to CNMARC. The relation schema and its realization algorithm of XMARC document have been studied specifically. Finally that new data of XMARC document has been got voluntarily by the data transmission from MARC text file to XMARC document.

Keywords Digital library, XMARC metadata, Relation schema

1 引言

数字图书馆系统建设中面临着以下迫切需要解决的问题:一方面 MARC 的数据大量存在,另一方面 MARC 自身存在的不足又在一定程度上制约其进一步的发展。基于 XML 的 CNMARC 实际上是一个面向 WWW 的开放的书目数据格式,它的出现是图书馆的重要发展机遇,将使数字图书馆系统在网络环境下呈现勃勃生机^[1,2]。为此,作者提出 XMARC 的新概念,认为 XMARC 在信息描述、信息交换、信息检索、信息发布等方面弥补了 MARC 的不足:(1)XMARC 在信息描述方面继承了 MARC 的所有优点, MARC 中包含的所有字段和子字段都体现在 XMARC 的元素或子元素中;(2)XMARC 依赖的语言基础是具有极好保值性的 XML,又具有 MARC 的数据可变长的优点;(3)XMARC 利用 XML 标记提供了很好的信息检索支持,可以通过 XMARC 数据库实现复杂检索;(4)XMARC 信息交换的灵活性使它与其它系统、其他 DC 等元数据的交换更加方便,这对于数据的馆际共享和国际数据的交换奠定了基础。

据此,本文根据 CNMARC 元数据设计了基于 XMARC 资源描述框架的 XML DTD,具体研究了 XMARC 文档的关系存储及其实现算法,最后通过 MARC 的数据转换实现了 XMARC 文本数据的自动获取。

2 基于 XMARC 资源描述框架的 DTD 设计

数字图书馆元数据 MARC 的现实含义为一条书目信息。本文定义框架 DTD 是关于 XMARC 资源描述框架的 DTD^[3],该文档类型定义把 MARC 字段的字段说明作为文档元素字段的一个属性,其属性名为“字段说明”,把子字段的字段说明作为文档元素子字段的一个属性,其属性名为“子字段

说明”。

上述 DTD 作为一种资源描述框架,能够将多种形式的 XMARC 统一起来。定义如下:

```
<?xml version="1.0" encoding="GB2312"?>
<!ELEMENT MARCS (MARC *)>
<!ELEMENT MARC (头标区, 字段+)>
<!ELEMENT 头标区 (#PCDATA)>
<!ELEMENT 字段 (子字段 *)>
<!ATTLIST 字段 字段说明 CDATA #REQUIRED>
<!ATTLIST 字段 tag CDATA #REQUIRED>
<!ATTLIST 字段 indicator1 CDATA #IMPLIED>
<!ATTLIST 字段 indicator2 CDATA #IMPLIED>
<!ELEMENT 子字段 (#PCDATA)>
<!ATTLIST 子字段 子字段说明 CDATA #REQUIRED>
<!ATTLIST 子字段 subtag CDATA #REQUIRED>
```

具体设计方法为:(1)对字段和子字段之间关系及说明的处理。该 DTD 定义的元素“字段”可以包含子元素“子字段”或纯文本数据(主要针对于不包含子字段的字段,例如字段001和字段005),子元素“子字段”可以在父元素“字段”中出现1到多次;(2)对字段和子字段标识符的处理。该 DTD 定义的属性缺省值为“#REQUIRED”而非“#FIXED”,含义为在 XML 文件中必须为这个属性给出一个属性值,但这个值不是固定的。(3)对字段指示符的处理。该 DTD 定义的属性缺省值为“#IMPLIED”而非“#FIXED”,其含义是不强行要求在 XML 文件中给该属性赋值。(4)对头标区的处理。在 MARC 中它是必备且不可重复的,即其在 MARC 中出现且只能出现一次, MARC 中头标区的前五位代表的是记录长度,这里以空格符填充。(5)根元素 MARCS 及其子元素 MARC 的处理。

王兰成 教授,博导,主要研究领域为数据库与信息处理、数字图书馆、信息系统与信息技术。

这种形式 DTD 的根元素为 MARCS, 包含一子元素 MARC, MARC 子元素是可重复的, 其元字符为“*”, MARC 元素包含两个子元素: 头标区和字段。

3 XMARC 文档的关系存储

将 XMARC 文档存储到关系数据库中分为三个步骤: DTD 规范化, 然后生成关系数据库模式, 最后将 XMARC 文档中的数据存储到关系数据库中^[4]。图1是 XMARC 文档关系存储系统的处理过程, DTD 经过规范化产生一个映射关系, 然后根据这个映射关系将 XMARC 文档中的数据装载到关系数据库中。

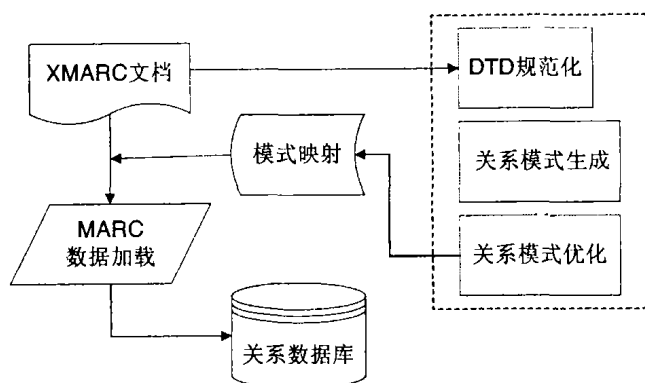


图1 XMARC 文档关系存储系统的处理过程

图2是 XMARC 文档的框架 DTD 图, 它反映了上述 XMARC 文档的层次结构。

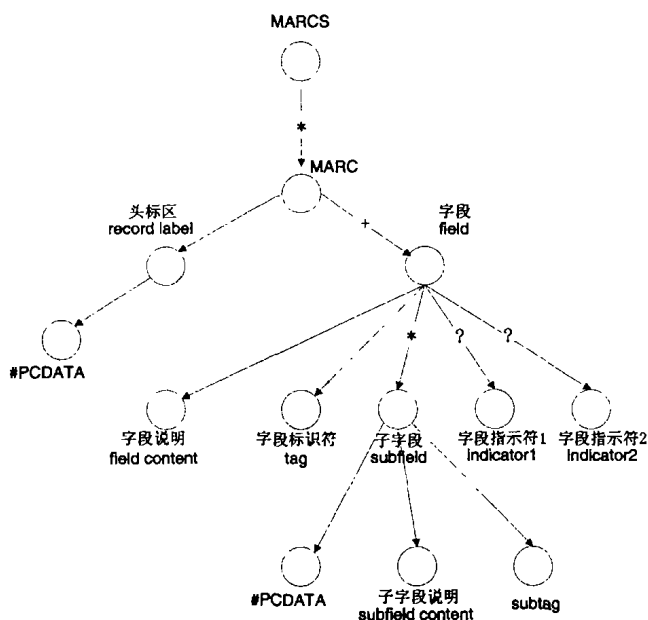


图2 XMARC 文档的一种 DTD 图

该 DTD 图中的偏序关系在图2中可以表示成父子联系, 其四种联系是: 直接联系 (如字段和 tag 之间的联系)、可空联系 (如字段和 indicator1 之间的联系)、+ 联系以及 * 联系。模式分解时采用深度优先的策略, 将树的叶子节点作为一个关系模式的属性。需要特殊处理的是 + 和 * 的一对多联系, 此时将子节点作为一个独立的子树进行模式分解, 并在该子树分解的模式中带上上层模式的 ID, 其分解算法如下。

算法1 根据 DTD 图, 将 XMARC 文档存储为关系模式。

输入: XMARC 文档的 DTD 图, 关系模式集合 U, MARCS (入度为零) 作为 d 第一入口节点。

输出: 关系模式集合 U。

步骤:

- 1 Begin
- 2 首次执行令 $U = \Phi$;
- 3 构造关系模式 R_1 , 根据入口点的名字构造标识符 (入口点名字) + id₁;
- 4 以深度优先顺序遍历所能到达的节点, 依次取出这些点的名字 $r_2, r_3, \dots, r_m$;
- 5 若能到达的所有点都已遍历, 转8;
- 6 直到遇到 + 联系或 * 联系为止;
- 7 取该 + 联系或 * 联系的子节点为新入口点递归调用本算法;
- 8 执行3;
- 9 构造关系模式 $R((\text{入口点名字}) + \text{id}, r_2, r_3, \dots, r_m)$, 将 R 加入 U ;
- 10 End

算法1以递归调用实现, 它给出了 XMARC 文档关系模式分解的基本算法。据此, 图2可以分解为如下关系模式:

```
MARCS (MARCSID: INTEGER, MARC: STRING);
MARC (MARCID: INTEGER, RECORD LABEL: STRING(24),
      FIELD: STRING);
FIELD (FIELDID: INTEGER, FIELD CONTENT: STRING, TAG:
       STRING(3), INDICATOR1: STRING(1), INDICATOR2:
       STRING(1), SUBFIELD: STRING);
SUBFIELD (SUBFIELDID: INTEGER,
          SUBFIELD CONTENT: STRING,
          SUBTAG: STRING(1));
```

通过以上工作, 将 XMARC 资源描述框架 DTD 生成关系模式, 从而实现了 XMARC 文档的关系存储。

4 MARC 对 XMARC 的信息映射

XMARC 可以在继承 MARC 优点的基础上弥补 MARC 的某些不足, 具有广阔的发展前景, 然而在实践中我们必须面临这样一个问题: 就目前来看, 绝大部分的书目数据采用的仍然是传统的 MARC 文本, 要把 XMARC 真正地应用于实践就必须考虑 MARC 文本到 XMARC 文档的数据映射问题。下面将给出由 MARC 文本到 XMARC 文档的转换算法2。

算法2 MARC 到 XMARC 的数据转换。

算法2首先取出1条 MARC 记录, 分别截取它的头标区、目次区和数据区的内容, 依次装配标识符, 生成 XMARC 的记录数据存储。

输入: MARC 文本。

输出: XMARC 文档。

变量说明: marcs 表示 n 条 marc 记录; xmarcs 表示 n 条由 marcs 转化而来的 xmarc 记录; onemarc 表示一条 marc 记录; toubiao 表示一条 marc 记录的头标区; mucu 表示一条 marc 记录的目次区; shuju 表示一条 marc 记录的数据区; onemucui 表示一个字段的目次区数据; oneshuju 表示一个字段的的数据区数据; zdname 表示字段说明; zddname 表示子字段说明; tag 表示字段标识符; subtag 表示子字段标识符; zsf1 表示字段指示符1; zsf2 表示字段指示符2。

步骤:

- 1 Begin
- 2 从指定文件中读取 marc 文本 $\rightarrow$ marcs;
- 3 xmarcs = "<?xml version=" + "1.0" + " encoding=" + "gb2312" + ">" + "<marcs>";
- 4 如果 len(marcs) $\neq 0$ 转5, 否则转22 (截取一条 marc 记录);
- 5 从 marcs 中截取一条 marc 记录 $\rightarrow$ onemarc;
- 6 从 onemarc 中截取头标区 $\rightarrow$ toubiao, 从 onemarc 中截取目次区 $\rightarrow$ mucu, 从 onemarc 中截取数据区 $\rightarrow$ shuju;
- 7 onexmarc = "<marc>"; onexmarc = onexmarc + "<头标区>" + toubiao + "</头标区>";
- 8 如果 len(mucu) $\neq 0$ 转9, 否则转19 (截取一条 marc 记录中的字段);
- 9 从 mucu 中取前12个字符 $\rightarrow$ onemucui;
- 10 根据 onemucui, 从 shuju 中截取到当前字段的数据内容 $\rightarrow$ oneshuju, 从 onemucui 截取当前字段的字段标识符 $\rightarrow$ tag, 根据 tag 从字段说明表中得到当前字段的字段说明 $\rightarrow$ zdname;
- 11 从 oneshuju 截取当前字段的指示符1 $\rightarrow$ zsf1, 从 oneshuju 截取当前字段的指示符2 $\rightarrow$ zsf2;
- 12 onemarc = onemarc + "<字段 字段说明=" + zdname + " tag=" + tag + " indicator1=" +

```

"" + zsf1 + "" + "indereator2=" + "" + zsf2 + ""
+ ";";
13 如果 len(oneshuj) ≠ 0 转 14, 否则转 18(截取当前字段中的子字
段);
14 从 oneshuju 中截取出子字段标识符→subtag, 根据子字段标识符和
子字段标识符从字段说明表中得到当前子字段的字段说明→zsd-
name, 根据子字段标识符从 oneshuju 中得到该子字段的内容→
zsdshuju;
15 onemarc=onemarc+"<子字段 子字段说明=" + "" + zsdname
+ "" + " subtag=" + "" + zdsbf + "" + ">" + shuju
+ "</子字段>";
16 oneshuju=oneshuj-subtag-zsdshuju, 转 13;
17 onemarc=onemarc+"</字段>";
18 mucu=mucu-onemuci, 转 8;
19 onexmarc = onexmarc + "</marc>";
20 xmarcs=xmarcs+onemarc;
21 marcs=marcs-onemarc, 转 4;
22 xmarcs=xmarcs+"</marcs>";
23 将 xmarcs 写入到指定文件;
24 End.

```

算法2是对有限长度的文本数据的线性转换, 头标区有固定的24位长, 当数据区内容确定后, 目次区也有确定的12N+1长(N为实际数据字段数), 根据字段标识符从字段说明表中得到相应的字段说明。实验示例如图3。结果表明, 该算法能正常执行, 数据转换正确。

5 XMARC 数据挖掘实验与结论

以主题标引为基础的主题检索是数字图书馆发展的必然趋势, 也是统一检索和标准化建设的需要, 在 XMARC 元数据上进行主题信息的数据挖掘则是一项重要内容。作者设计了最大匹配最小推进模式的中文抽词方法, 即当在词表中按最大匹配的原则匹配到长度的词后, 在以首字命中记录中找到最小匹配的词, 并按最小匹配词的长度移动指针^[5]。对 XMARC 分别实验如下三种方法: (1) 用一般禁用词表及最大匹配整词推进的模式进行匹配的词典主题自动标引; (2) 用一般禁用词表及最大匹配单字推进的模式进行匹配的词典主题自动标引; (3) 本文提出的用具有位置信息的禁用词表及最大匹配最小推进的模式进行匹配的词典主题自动标引。准备 a. 5000 条含标题和文摘(平均每条 40.05 字)的标引源, b. 5000 条全部是标题(平均每条 9.02 字)的记录, 在两组数据中随机抽出 500 条较长(平均 37.6 字/条)记录和 500 条较短(平均

9.083 字/条)记录进行手工标引, 分别用三种方法处理, 计算完全相似率和基本相似率(标引结果含同义词)。三种方法均在同等硬软件环境下运行^[6], 结果如图4。

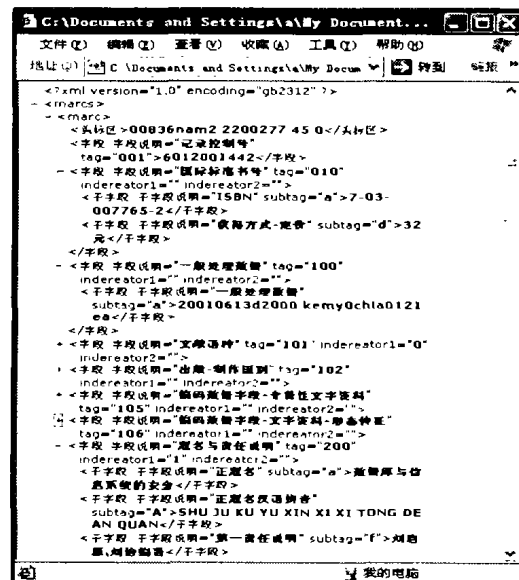
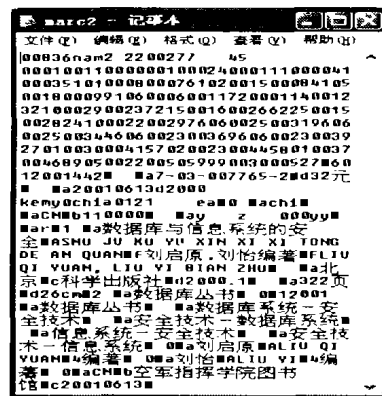


图3 MARC 到 XMARC 的转换实验示例

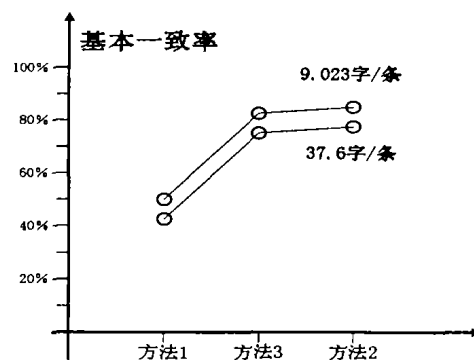
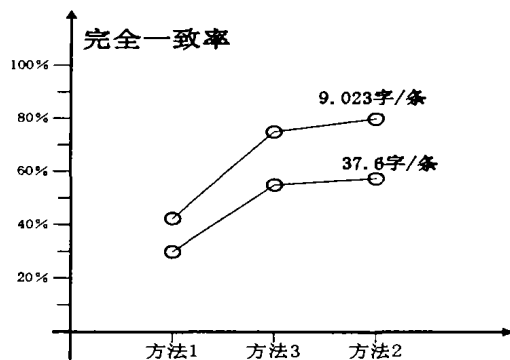


图4 XMARC 主题信息标引实验示例

显然, 方法(2)完全一致率和基本一致率都是最高的, (1)的一致率都是最低的, (3)一致率中等, 更靠近(2), 在时间和质量的综合性能上优于以(1)、(2)为代表的传统标引方法。

MARC 得到广泛应用的同时, 其局限性随着应用背景的变化而逐渐暴露出来。本文基于 XML 提出 XMARC 概念并进行框架 XML DTD 方案的设计, 具体研究了 XMARC 文档的关系存储及其实现算法, 最后基于 MARC 文本文件向 XMARC 文档的数据转换给出了实现算法。该研究对于现有 CNMARC 数据在数字图书馆中的利用具有重要的现实意义。

义。

参考文献

- W3C XML 主页. <http://www.w3.org/XML/>
- Weibel, Stuart. The State of the Dublin Core Metadata initiative. D-lib Magazine, April 1999. <http://www.dlib/april99/04weibel>
- 王兰成, 冯文杰. 两种 CNMARC 的 XML DTD 信息描述机制的研究与比较. 中国图书馆学报, 2004, 30(1): 70~73
- 何震瀛. XML 数据的关系存储. 计算机科学, 2002(8. 增刊): 24~26

5 王兰成,等. PLS:一种基于信息自动标引的最小推进分词算法及其实现. 计算机科学,2002(8. 增刊):24~26

6 田梅. 档案机读目录 XML 描述及其主题信息自动标引的研究: [硕士学位论文]. 上海:南京政治学院上海分院信息管理系,2004

(上接第133页)

发挥其优越性时(每个段只包含一个算子),我们的策略仍按照贪心法进行调度.因此可以说我们的策略比单纯的采用循环或者先入先出策略时,占用的系统内存要少.

文[8]中已经证明了 Chain 策略与最优调度策略的误差为 n ,其中 n 表示连续查询的总数.在我们提出的合并查询中(假设它也由 n 个查询组成,且数据流缓冲区中没有元组),某一时刻共有 m 个元组正被这个合并查询进行操作,也就是说这 m 个元组已经被第一个共享表达式的算子处理过.我们已经证明了在合并查询中,1个元组的大小在过程图中的最大值变为 n ,由此可得任意时刻合并查询中元组大小的最大值为 $m \times n$.由于没有策略能够不占用内存,因此这个值也可以被认为是与用最优策略调度时内存占用量的最大误差.而与 Chain 策略的误差 n 相比,我们的策略在最坏情况下多出了 $m \times n - n$ 个元组大小的空间.然而,由于这 m 个元组都来自数据流缓冲区,而这个缓冲区为 n 个查询所共享,因此,与不进行资源共享的策略比起来,共享策略在数据流缓冲区部分就已经节省了 $m \times (n-1) = m \times n - m$ 个元组的空间.将上面两个值相减得 $m - n$,它表示的是假设只有数据流被共享时,最坏情况下我们的共享资源策略比不共享资源的 Chain 策略多占用的元组数.

很明显,当系统发生数据爆发时(数据流缓冲区及共享子表达式部分有大量的元组),我们的策略相比 Chain 可以非常明显地降低系统内存的占用.同时,我们还可以规定当合并查询正在处理的元组数达到 n 时,暂停处理数据流中的数据.这样就可以保证我们的策略在任意时刻都比不能共享资源的 Chain 策略占用更少的内存.

6 模拟试验分析

我们模拟了两个简单的连续查询 Q_1 和 Q_2 , Q_1 的各个算子在过程图中的点 (t, s_i) 分别为 $(0, 1)$, $(1, 0.7)$, $(2, 0.4)$, $(3, 0.3)$ 和 $(4, 0)$, Q_2 的算子所对应点的坐标分别为 $(0, 1)$, $(1, 0.7)$, $(2, 0.3)$, $(3, 0.2)$ 和 $(4, 0)$.其中 Q_1 和 Q_2 共享相同的数据流 S 以及第一个算子 O_1 .我们假设初始状态共享数据流中有1000个元组(对 Chain 来说则是2000个),并令每一个算子处理一个元组的执行时间都为1毫秒.

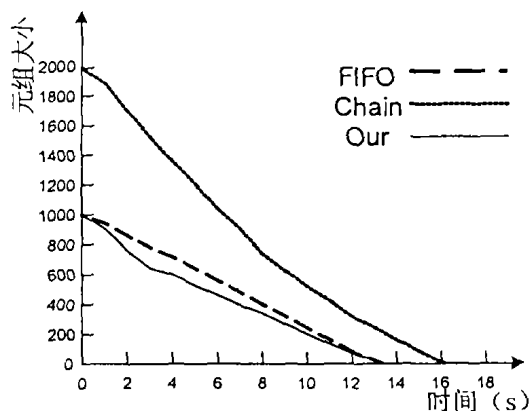


图5 试验结果

实验中,我们对先入先出,Chain 以及我们的策略进行了编程实现,图5显示了应用不同策略时元组数随时间变化的情况,这里我们假设先入先出策略在执行各个查询的后续操作部分时采用顺序执行的办法,即顺序完成各个查询.

从试验结果中我们可以看出,我们的策略要比采用先入先出和 Chain 策略具有更好的效果.

结论与展望 本文针对现有策略中没有考虑资源共享问题的情况给出了一种数据流应用系统中查询算子的调度策略.这种策略继承了 Chain 策略的优点,并在一定程度上减小了引入资源共享问题所带来的误差.模拟的试验证明,与不能共享资源的 Chain 策略和可以共享资源的 FIFO 策略相比,我们给出的策略在数据爆发时能够更有效地减小内存占用.

在实际应用中,可以将我们的算法与 Chain 相结合,对能够进行资源共享的查询应用我们的策略,而对于不能进行资源共享的查询则应用 Chain 策略,这样可以更进一步提高系统的执行效率.

在将来的工作中我们会在真实环境中进一步检验我们的策略,并将这种策略扩展到多数据流查询的情况,比如含有连接操作的查询.

参 考 文 献

- Babcock B, Babu S, Datar M, Motwani R, Widom J. Models and Issues in Data Stream Systems. In: Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symp on Principles of Database Systems, June 2002. 1~16
- Babu S, Widom J. Continuous Queries over Data Streams. SIGMOD, 2001. 30(3): 109~120
- Abadi D J, Carney D, Cetintemel U, et al. Aurora: a new model and architecture for data stream management. In VLDB Aug. 2003. 120~139
- Chandrasekaran S, Cooper O, et al. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In: Proc. of CIDR Conf. Monterey, California, Jan. 2003
- Motwani R, Widom J, et al. Query processing, approximation, and resource management in a data stream management system. In: Proc. of First Biennial Conf. on Innovative Data Systems Research (CIDR). Monterey, California, Jan. 2003. 245~256
- Carney D, Cetintemel U, Cherniack M, et al. Monitoring streamsa new class of data management applications. In: Proc. 28th Intl. Conf. on Very Large Data Bases. Hong Kong, China, Aug. 2002
- Babcock B, Babu S, Datar M, Motwani R, Thomas D. Operator Scheduling in Data Stream Systems; [Technical report]. Stanford University STREAM Group, 2004. Available at: <http://www-db.stanford.edu/stream/>
- Babcock B, Babu S, Datar M, Motwani R. Chain: Operator Scheduling for Memory Minimization in Data Stream Systems. In: Proc. of ACM SIGMOD Intl. Conf. on Management of Data. San Diego, California, June 2003