列存储数据区级压缩模式与压缩策略选择方法

王振玺[1)] 乐嘉锦[1)] 王 梅[1)] 刘国华[1),2)]

[1)](东华大学计算机科学与技术学院 上海 201620)

[2)](南京大学计算机软件新技术国家重点实验室 南京 210093)

摘 要 压缩技术是列存储数据管理的重要研究内容之一.目前多数方法对同一列数据使用单一压缩方法进行压缩,忽略了数据的局部分布特性,极大地影响了压缩性能.该文提出一种区级压缩模式,并在此模式下提出基于学习的压缩策略选择方法.首先该文将数据列进一步划分为区,并分别定义相邻区信息与区所在列的统计信息为参照信息,进而通过学习参照信息与当前区之间的相似性和差异性进行策略推荐.最后该文对区进行局部学习从而对推荐压缩策略进行修正,保证压缩策略的有效性.在数据仓库基准数据集 SSB 上的实验结果验证了该文方法的有效性.

关键词 列存储;数据压缩;区级压缩模式;压缩策略选择

中图法分类号 TP311 **DOI 号**:10.3724/SP.J.1016.2010.01523

Sector-Based Compression and Compression Strategy Selection Method for Column Stores

WANG Zhen-Xi[1)] LE Ja-Jin[1)] WANG Mei[1)] LIU Guo-Hua[1),2)]

[1)](*School of Computer Science and Technology, Donghua University, Shanghai* 201620)

[2)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)

Abstract Compression technology is an important research field in column-oriented management system. However, most previous compression techniques for column-oriented data use same algorithm for all columns, ignoring the local distribution of data, which greatly degrade the compression performance. This paper proposes a sector-based compress pattern, under such pattern further provides a novel learning-based compression strategy selection method for column stores. First, data column is divided into sectors in the method. The neighbor sector information and the statistic information of the column with the given sector respectively are extracted as two references. Then by learning the similarity between the reference and the given sector the recommended compression strategy can be obtained. Finally, the recommended compression strategy is improved by partly learning the given sector to guarantee the effectiveness of it. The experimental results on data warehouse benchmark data set SSB testify the effectiveness of the proposed method.

Keywords column store; data compression; sector-based compression; compression strategies

1 引 言

随着数据仓库技术的不断发展,数据仓库中包含的数据量急剧增加,对传统的行存储模式的查询提出了挑战.为了提高读优化(read-optimized)系统的性能,人们开始考虑一种与传统行存储不同的存储方式-列存储.列存储技术是将数据表以列为单位

收稿日期:2010-06-11.本课题得到国家自然科学基金(60773100)资助.**王振玺**,男,1985 年生,硕士研究生,主要研究方向为列存储、数据压缩.E-mail:wzx@mail.dhu.edu.cn.**乐嘉锦**,男,1951 年生,教授,博士生导师,主要研究领域为数据库与数据仓库、软件工程理论与实践.**王 梅**,女,1980 年生,博士,主要研究方向为数据库与多媒体.**刘国华**,男,1966 年生,教授,博士生导师,主要研究领域为隐私保护、文档复制检测、*k*-匿名隐私保护模型下的不确定性数据管理、面向数据的业务过程管理.

进行存储，数据表记录中的同一属性值被存储在一起.在进行查询的时候，列存储数据仓库系统只需要将需要的列读入内存，一定程度上减少了读入的数据量，使得系统的查询效率得到提高[1].然而，数据仓库需要处理的数据量是非常庞大的，这造成查询时大量的I/O.由于CPU处理与磁盘访问发展的不平衡，造成了I/O成为了查询的瓶颈.因此，减少I/O的次数能显著提高查询的效率.而数据压缩则能在一定程度上减少I/O的次数，因此，数据压缩成为了一个研究的热点.对于列存储来说，数据具有相同的数据类型，增加了相邻数据之间的相似性，使得列存储系统和传统的行存储系统相比具有更好的压缩效率.

数据压缩一直以来是数据管理领域长期关注的研究问题，目前存在众多的压缩方法，如Huffman编码[2]、算术编码[3]、字典编码(Dictionary-based compression)、空值抑制(Null Suppression)[4]、游程编码(Run-Length Encoding)[5]、LZ系列编码等[6].Oracle，DB2，SQL Sever2008等基于行存储的数据库管理系统主要采用字典编码、空值编码等压缩方法来实现压缩.C-Store[7-8]等列存储系统则主要采用游程编码、字典编码、空值编码等压缩方法.研究表明：不同压缩方法对同种数据类型的压缩效果是不同的，同样相同压缩方法对不同类型数据的压缩效果也是不同的.因此，对不同的数据类型采用不同的压缩方法十分必要.然而目前在压缩策略选择方面的研究非常少.Abadi等人[7-8]通过实验提出了一个决策树，通过该决策树进行压缩策略选择，然而这种压缩是建立在整个列上面的，即该方法将一个列的数据采用同一种压缩方式进行压缩.但是该方法忽略了局部数据在分布上存在着差异性.如图1所示，一个列中第101个和第102个区的数据

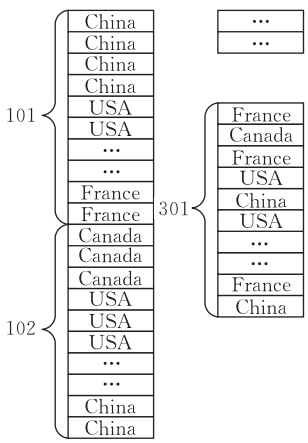

图1 列中的数据

的特点相似，可以采用游程编码，但是第301个区中的数据与上面两个区的差异就比较大，如果采用游程编码，压缩效率就非常低.对该区采用字典编码则可以很好地解决这个问题.由此看来在较小的粒度上面进行压缩可以提高压缩率.然而，为每个区进行学习时间复杂度高，并不可行.

针对上述问题，本文提出一种区级压缩模式下的基于学习的压缩策略选择方法.首先本文将数据列进一步划分为区，并分别定义相邻区信息与区所在列的统计信息为参照信息，在此基础上提出两种不同的压缩策略选择方法：基于相邻区的参照学习方法通过定义连续区之间的相似因子来判定连续两个区中数据分布的相关性和差异性，从而预测当前区中数据的分布特点，得到推荐的压缩策略；基于统计参照区学习方法通过估计当前区和区所在列的整体分布相似性进行策略推荐.在此基础上，本文对区进行局部学习从而对推荐压缩策略进行修正，保证压缩策略的有效性.上述两种方法都有效地利用数据的局部分布特点进行压缩，并降低了单独为每个区进行学习的额外开销.在数据仓库基准数据集SSB上与C-Store的方法进行比较，经过实验验证本文的方法较C-Store决策树的压缩率有较大的提高.

本文第2节介绍相关工作；第3节介绍本文定义的一些变量和概念；第4节介绍基于学习的压缩策略选择方法的详细内容；第5节描述本文提出的两种方法在数据仓库基准数据集SSB上进行的实验，并对实验结果进行详尽的分析和说明；最后一部分是总结和展望.

2 相关工作

列存储思想的出现可以追溯到1975年[9].后来，Wong等人提出了移项文件[10]的技术用于解决对大型科学数据库的检索速度问题.他提出将一个关系的不同列的值存放在不同的文件中，并且对这些列值进行了压缩以节省空间.这是列存储的雏形.Copeland等人提出了DSM(Decomposition Storage Model)[11]存储模型的概念.Ailamaki在DSM存储模型的基础上提出了PAX(Partition Attributes Across)[12]技术，这种技术通过将数据块分割为小的数据区，在每个数据区中存储一个属性的值，提高了数据的查询性能.Ramamurthy等人提出了碎片镜像(Fractured Mirrors)技术[13]，这种方法实质上是一种行存储和列存储的折中，通过在一个系统中

保存两种版本(NSM 存储模型和 DSM 存储模型)的数据副本来提高查询性能. 后来,Abadi 等人提出了 C-Store[7-8]存储架构并且实现了一个实验性质的系统 C-Store,这个系统的特点是上层查询执行仍然采用行存储即元组的方式执行而在磁盘上采用列存储的方式存储数据,在上层和底层之间有一个称为 Tuple Mover 的组件用于上层和下层数据之间的迁移. 由于列存储数据是以属性或列为单位进行存储的,数据具有相同的数据类型,使得列存储系统和传统的行存储系统相比具有更好的压缩效率. 压缩技术成为列存储数据管理的重要研究内容之一.

适用于数据库上面的压缩方法有很多,主要有空值悬挂[4]、游程编码[5]、位图编码、字典编码[4]以及 LZ 系列的字典编码[6]等,如表 1 所示. 从压缩粒度的角度来说,目前压缩主要在以下 5 种不同的粒度中进行:表级、块级、元组级、属性级以及比特级压缩[1]. 表级粒度下的压缩方案,具有代表性的是 IBM 公司的 DB2 9(又名 DB2 Viper),DB2 9 中采用全表级数据库压缩方案,但是全表级压缩方案使得在解压记录时需要大量的 I/O. 其中元组级的压缩方案很少见. 块级粒度下的压缩方案就比较多了. Eggers,Shoshani 提出了一种对多维稀疏数组进行数据页级压缩的方法,其具有 forward mapping 和 backward mapping 这种 mapping complete 性质;Goldstein 等人给出了基于 frame of reference 的数据页级压缩算法,该算法找出每一页中元组之间的共同信息,称之为 frame of reference,其它数据参照此进行压缩,将多个压缩数据页结合起来就形成了文件级的压缩;Oracle 的压缩算法采用了基于字典的压缩方法[14],其将字典和数据放在同一页中,但是这种方法的压缩效率相对全表级压缩来说,压缩率较低. 比特级粒度下的压缩方案也有很多,但是应用的领域有限制. Wong 等人给出了基于 Bit 级压缩的 Bit TranspositionFile(BTF)系统[15],该系统针对科学统计数据库中属性的候选集有限的特点,将属于相同列的 Bit 存放在同一数据块中实现压缩;而对于那些不能压缩的属性,BTF 仍然按照 Bit 列存储. BTF 只考虑了单表的查询操作,没有给出连接操作.

除了在行存储上面取得了很多的研究成果之外,列存储上面的数据压缩研究也取得了一定的成果[7-8,16-17]. Abadi 等人提出一种按属性列存储、读优化的、压缩的数据库管理系统 C-Store[7-8],主要采用游程编码、字典编码、位图编码、LZ 系列编码等压缩方法. 以哈尔滨工业大学的李建中教授为首的研究小组针对科学与统计数据库的特点提出了一个比特级压缩方案[17]. 该方案对于值域有限的属性,采用 K-of-N+Unary 进行压缩编码,并将编码结果按照 Bit 列存储,实现压缩;对于主键属性采用 Double-Key-Btree 按属性列存储;对于其它属性则按照 TupleID-Btree 按属性列存储. 但是该压缩方案将一个数据库表拆分为两个数据库表的方式使得插入、删除和更新操作的执行变得十分复杂,使得该拆分压缩方案的使用范围过于狭窄.

在压缩技术不断发展的同时,对压缩策略选择的研究也在不断的发展. Abadi 等人提出了的数据库管理系统 C-Store[7-8]. 它通过建立一个决策树来判定一个列采用何种压缩算法. 但是这种方法忽视了数据在局部上的特殊性,过于粗糙,造成压缩率相对比较低.

表 1　常见的压缩算法

序号	压缩算法	适合的数据类型
1	字典编码	字符(串),数值,日期
2	游程编码	字符(串),数值,日期,布尔
3	位图编码	字符(串),数值,日期,布尔
4	空值编码	字符(串),数值,日期,数值,文本
5	整数编码	整数
6	LZ 编码	字符串,文本

3　定义和符号

给定关系模式 $R=\{A_1, A_2, \cdots, A_n\}$,其中 n 表示关系的目的度,A_i 表示第 i 个属性列. 在列存储中,任意一个列 A_i 中的数据在逻辑上对应一个数据段 $S_i \in S$. 本文中将段分为若干个区,区是一系列连续块的集合. 第 i 个属性列对应的段记为 $S_i=\{e_1^i, e_2^i, \cdots, e_n^i\}$,其中 e_{j-1} 和 e_j 是两个连续的区. 区由 16 个连续的数据库块构成. 区中的第一块是区的控制块,保存区中所有块的控制信息. 区的结构如图 2 所示.

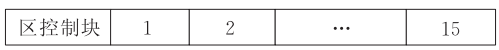

图 2　区的结构

区中存放着连续的列值,每一个列值称为一个项(item). 本文为每一个区定义一组统计信息,记为集合 $T=\{r,s,a,d,n,c,l\}$,其中 r 表示第 i 个区中 item 的数目,s 表示区中相同值的数目,a 表示区中相同值的总的 item 条数,d 表示区中不同值的数

目，n 表示区中空值的数目，c 表示区中相同值连续的平均数目，l 表示区中 item 的平均长度. 基于上述统计信息，本文为每一个区定义了一组统计量 $q_i=\{q_i^1,q_i^2,q_i^3,q_i^4,q_i^5\}$，$q^i$ 描述了第 i 个区的数据分布的一些特点，详细如下：

定义统计量 1：q_i^1 表示区中相同值所占的百分比，$q_i^1=a\times 100/r$.

定义统计量 2：q_i^2 表示区中空值记录所占的百分比，$q_i^2=n\times 100/r$.

定义统计量 3：q_i^3 表示区中相同值连续的平均数目，也就是 $q_i^3=c$.

定义统计量 4：q_i^4 表示区中不同值的数目，也就是 $q_i^4=d$.

定义统计量 5：q_i^5 表示区中 item 的平均长度，也就是 $q_i^5=l$.

例如，一个区中共有 $r=1000$ 条的记录，其中相同值的数目 $s=20$，相同值的总记录条数为 $a=960$，区中不同值的数目 $d=30$，区中空值的数目 $n=10$，区中相同值连续的平均数目 $c=5$，区中 item 的平均长度 $l=10$，那么，$q_i^1=960\times 100/1000=96$，$q_i^2=10\times 100/1000=1$，$q_i^3=5$，$q_i^4=30$，$q_i^5=10$.

令压缩算法集合 $m=\{m_1,m_2,\cdots,m_n\}$，n 为算法数量，m_i 中的压缩算法分别取自表 1 中的算法. 定义函数 $f:T\rightarrow m_T$ 表示根据区的统计信息 T 得到适合区中数据分布的最优压缩算法 m_T.

4 区级压缩模式与基于学习的压缩策略选择方法

如前所述，在一列上采用单一的压缩算法进行压缩忽略了数据的局部分布特点，因此本文提出区级压缩模式. 所谓区级压缩模式是指压缩的基本单位是区. 不论采用什么类型的压缩算法，这种算法针对的对象是一个区中的数据. 例如对于字典编码来说，字典的建立是在对区中的数据进行统计之后得到的，这个区和另一个区的字典的内容是不一样的，字典存放在区控制块中；对于位图编码来说，区中的不同值的数目决定了要采用的位图编码的向量的数目. 然而，列中数据分布的一致性和相似性决定了为每个区单独学习会造成巨大的时间和空间的浪费. 为此，本文提出基于学习的压缩策略选择方法，通过学习当前区与参照区之间的相关性和差异性，进行策略推荐，从而在得到较优压缩策略的同时减少时间复杂度.

4.1 基于相邻参照区的压缩策略选择

基于相邻参照区的压缩策略选择方法的基本思想是：局部连续区之间的数据分布一般情况下具有一致性和相似性，可以采用相同压缩方法. 因此，本文首先利用区的统计量估计相邻两区数据分布的相似性. 为此，这里定义连续区 r_i 和 r_{i+1} 的相似因子为 $S=\{S_1,S_2,\cdots,S_5\}$，

$$S_i^j=|q_i^j-q_{i+1}^j| \tag{1}$$

其中 $j\in\{1,2,\cdots,5\}$. 若 $\exists S_i^j\in S, S_i^j<\delta^j$ 则认为两个区近似，可以用相邻区进行策略推荐，可得到 $m_i=m_{\text{temp}}$，其中推荐压缩方案为 m_{temp}.

否则则认为两者不相似，需要对当前待压缩区进行局部学习，从而对压缩策略进行修正.

这里的 $\delta_i\in\delta$ 为一些临界值，不同的 q 对应的临界值是不同的，其最优值需要在实验中确定.

基于相邻参照区的压缩策略选择的算法总结如下.

算法 1.

```
Function Compression_Strategies(segment s)
输入：s 需要压缩的段
输出：压缩是否成功 (0：失败，1：成功)
1. 设置 i=1 为段 s 的第 1 个区的区号
2. 设置 m1=0，推荐压缩策略 mtemp=m1
3. while ei 不是最后一个区
4.    if i==1
5.    计算统计信息 Ti={ri,si,ai,di,ni,ci,li}
6.    mtemp=f(Ti)
7. else
8.    根据 mtemp，获得 ri,si,ai,di,ni,ci,li 中的相关信息
9.    根据式(1)计算对应的相似因子 Sj
10.   if ∃Sij<δj
11.      mtemp=mi-1
12.   else
13.      获得当前区的统计信息 Ti={ri,si,ai,di,ni,ci}
14.      mtemp=f(Ti)
15.   end if
16. end if
17. mi=mtemp
18. 按照 mi 的压缩方式对本区的数据进行压缩
19. i=i+1
20. end while
21. end Function
```

步 3～步 20 是算法的主要部分，实现整个列的压缩. 步 4～步 6 统计第一个区的统计信息，并获得第一个区的压缩算法. 步 7～步 17 处理列中其它区的信息；步 8 根据推荐压缩算法来推断需要统计的信息；步 9 计算出相似因子，如果相似，则本区的压

缩算法就采用推荐的压缩算法，否则就重新统计区中的相关信息并得到压缩算法.

以图 1 中的数据为例，比较适合第 101 个区中的压缩算法为游程编码，那么第 102 个区的推荐压缩算法也就为游程编码，那么判断游程编码是否适合第 102 个区，只需通过统计第 102 个区中相同值连续的平均数目即可，如果不小于 4，则适合采用游程编码，否则需要重新进行学习，对推荐压缩算法进行修正. 根据图 1 所示，第 102 个区的数据可以采用游程编码. 以此类推，当到第 301 个区时，不满足游程编码的条件，因此这个区需要重新学习，从而得到第 301 个区的压缩方案为字典编码. 那么下一个区的推荐压缩方案为字典编码.

上面的 f 是通过统计量获得合适压缩算法，这里令 f=Get_Compression_Method，算法如下：

算法 2.

Function Get_Compression_Method(统计信息 T_i)

输入：当前区的统计信息 $T_i=\{r_i,s_i,a_i,d_i,n_i,c_i\}$

输出：压缩策略

1. 推荐压缩策略 m_{temp}=不压缩
2. 计算 $q_i=\{q_i^1,q_i^2,q_i^3,q_i^4,q_i^5\}$的值
3. if 区中的数据是布尔类型
4.　if 数据是有序的
5.　　m_{temp}=游程编码
6.　else
7.　　m_{temp}=位图编码
8.　end if
9. else if $q_i^2>\sigma_1$
10.　　编码方式为空值编码
11. else if $q_i^1>\sigma_5$
12.　　if $q_i^3\geq\sigma_2$
13.　　m_{temp}=游程编码
14.　　else if $q_i^4\leq\sigma_3$
15.　　m_{temp}=位图编码
16.　　else
17.　　m_{temp}=字典编码
18.　　end if
19. else if 区中的数据是整数类型
20.　　m_{temp}=整数压缩编码
21. else if $q_i^5>\sigma_4$
22.　　$m_{\text{temp}}=LZ$
23. else
24.　　m_{temp}=不压缩
25. end if
26. Return m_{temp}
27. end Function

算法中的 $\sigma_1,\sigma_2,\sigma_3,\sigma_4,\sigma_5$ 是用来估计压缩算法的阈值，其具体取值在实验部分给出. 例如算法第 9 步 $q_i^2>\sigma_1$ 表示区中空值大于一定阈值，则推荐压缩算法为空值压缩. 这里的空值编码是指对于 item 的值为空值时，数据库将不分配空间，只是给一个标记用来表示 item 的值为空. 整数编码采用的是文献[2]的算法. 算法的步 3～步 8 是处理布尔类型的数据，这是因为布尔类型的数据只有两个值，那么统计量 $q_i^1=100$，$q_i^4=2$ 或 $q_i^5=1$，因此需要特殊处理. 当数据是有序的情况下，则意味着区中的数据的结构为图 3 中的 3 种类型中的一个，因此，用游程编码是最合适的；当数据是没有排序的情况下，采用位图编码是比较合适的.

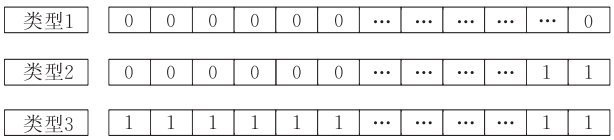

图 3　布尔类型数据在区中的结构

算法的步 9～步 10 是用来处理区中空值比较多的情况. 步 11～步 18 是用来处理区中公共值比较多的情况的. 其中步 12～步 13 用来处理区中的公共值大多是连续出现的情况的，因此采用游程编码；步 14～步 15 则是针对不同值相当少的情况的，同时也避免了数据是经过排序的情况，因此采用位图编码还是比较好的选择；在其它情况下，则采用字典编码，这就是步 16～步 17 的内容. 步 19～步 20 则处理数值类型中既没有太多的空值，又没有太多的相同值的情况下的压缩，这种时候采用整数压缩编码还是很好的选择.

4.2　基于参照列的压缩策略选择

基于参照列的压缩策略选择的基本思想是：数据在整个列上的分布具有一致性，因此可以用整个列的统计信息作为参照推断待压缩区的数据分布，从而进行策略推荐. 这里从待压缩的段中随机抽取若干个区(如段中区的数目的 1/20)作为样本，将这些抽取出来的区中数据的分布代表整个段中的数据的分布特点. 通过对这些抽取出来的区的数据进行统计，得到与统计信息相匹配的压缩算法，从而确定这个段的推荐压缩方式. 与该段中统计数据分布相似的区，将采用推荐的压缩策略压缩，否则将重新学习来确定压缩算法. 这里将随机抽取的区中的数据的相关统计信息记为 $T=\{r,s,a,d,n,c\}$，统计量为 $q=\{q^1,q^2,q^3,q^4,q^5\}$，这里定义了当前区 r_i 和样本区 r 的相似因子为 $S=\{S_1,S_2,\cdots,S_5\}$，

$$S_i=|q_i-q| \tag{2}$$

推荐压缩方案 $m_{\text{temp}}=f(T)$，其中 T 为抽取出来的区的统计信息.

若 $\exists S_i^j \in S, S_i^j < \delta^j$ 则认为当前区与列的分布近似，可以进行策略推荐，可得到 $m_j = m_{\text{temp}}$.

否则则认为两者不相似，需要对当前待压缩区进行局部学习，从而对压缩策略进行修正.

其中 $\delta_i \in \delta$ 为一些临界值，和 4.1 节相同.

那么通过算法 2 获得合适的压缩方法.

基于参照列的压缩策略选择的算法如下.

算法 3.

Function Compression_Strategies(segment s).

输入：s 需要压缩的段

输出：压缩是否成功 (0：失败，1：成功)

1. 设置 $m_1 = 0$，推荐压缩策略 m_{temp}
2. 从段 s 中随机挑选出约 1/20 的区
3. 获得这些区的统计信息 $T = \{r, s, a, d, n, c\}$
4. 得到推荐压缩算法 $m_{\text{temp}} = f(T)$
5. while s 中的数据没有压缩完毕
6. 根据 m_{temp}，获得 $r_i, s_i, a_i, d_i, n_i, c_i, l_i$ 中的相关信息
7. 计算相似因子 S_i
8. if $\exists S_i^j < \delta^j$
9. 采用 m_{temp} 进行数据压缩
10. else
11. 获得当前区的统计信息 $T_i = \{r_i, s_i, a_i, d_i, n_i, c_i\}$
12. 调用 $f(T_i)$ 来得到合适的压缩算法并压缩数据
13. end if
14. $i = i + 1$
15. end while
16. end Function

算法的步 2～步 4 主要是获得随机区中的数据并进行统计得到推荐压缩算法. 步 5～步 15 则是对每一个区的处理过程. 步 8～步 9 是对与段的数据分布相同的区的压缩算法采用推荐压缩算法，而与段的数据分布不相同的区的处理则是在步 10～步 13 中.

以图 1 中的数据为例，通过对抽取一部分随机的区的数据的分析，可以得出这个段的推荐压缩方案为游程编码. 在处理第 101 个区时，推荐压缩方案为游程编码，而第 101 个区的数据分布的特点满足游程编码的要求，因此不用进行学习. 当处理到第 301 个区时，统计分析的结果不满足右侧很难过编码的特点，因此第 301 个区需要重新学习，从而得到适合的压缩方案为字典编码，但是此时下一个区的推荐压缩方案仍为游程编码，而不是字典编码，这和算法 1 有着明显的区别.

5 实验验证

(1) 实验环境

Intel(R) Pentium(R) 4 CPU 2.80GHz，内存 DDR 512MB，缓存 L1：8KBytes，L2：12Kuops，L3：512KBytes. 软件环境操作系统为 Windows XP Professional Version 2002 Service Pack 3，开发环境为 Microsoft Visual C++6.0. 本文用 C 语言实现了基于学习的压缩策略选择方法，在此基础上，设计了两个实验，用来和 C-Store 的决策树进行相比，从压缩率和压缩所需要的时间两个方面进行对比.

(2) 数据集描述

本文实验的数据是采用数据仓库基准数据集 SSB 中的 *customer* 表的数据. 本文选择 *customer* 表的 *mktsegment* 列来进行测试. 这是因为在数据库使用的压缩方法中，字典编码、游程编码和位图编码是最常用也是最重要的 3 种压缩方法. 在 *mktsegment* 列中一共只有 5 个不同的值. 根据观察可以看出，如果基于 C-Store 决策树的方法，那么根据 C-Store 决策树分析，应该采用位图编码来对数据进行压缩. 而本文提出的基于学习的区级压缩模式下的策略选择方法则可以采用的压缩方案为字典编码、位图编码和游程编码. 因此本文提出的两种实现方法和基于 C-Store 决策树的方法就可以很好地在同样的数据中进行比较，由此看来 *mktsegment* 列有很强的代表性. 本实验中采用的区的大小为 16KB，可以存放大约 2000 个左右的数据. 本文分别对 660KB，1980KB，3300KB，4620KB，5940KB，7260KB，8580KB，9900KB，11220KB 9 种不同大小的数据进行了测试. 根据不同压缩算法的特点，本文方法中涉及的一些参数取值如下：$\sigma_1 = 30, \sigma_2 = 4, \sigma_3 = 4, \sigma_4 = 20, \sigma_5 = 20, \delta^1 = 5, \delta^2 = 5, \delta^3 = 1, \delta^4 = 3, \delta^5 = 5$.

(3) 实验结果与分析

本文从以下两个方面对 3 种不同压缩策略选择方法：基于学习的压缩策略选择方法、基于统计分析的压缩策略选择方法和基于 C-Store 决策树的压缩策略选择方法进行比较：

a) 压缩率比较. 通过在同一种数据类型(这里为字符串类型)不同数据量上进行比较，如图 4 所示. 图中的数据不包含槽的长度(每一个项(item)对应一个槽，每个槽的长度为 13B)，指的是每一个

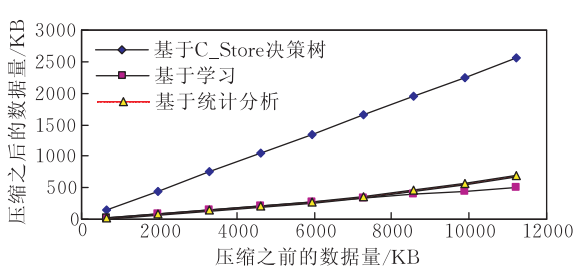

图 4 压缩效果图

item压缩之后的长度.

从图 4 可以看出，本文提出的两种压缩策略选择方法要比基于学习的区级压缩模式下的策略选择方法的压缩效果好. 这是因为，当数据量大的时候，数据的整体的特点虽然变得比较清晰，但是在局部的地方，数据的分布和整体还是有很大的区别. 并且随着数据量的增大，和整体数据具有不同分布的局部变得越来越多，这也就造成了这些局部的数据需要用更加合适的方法进行压缩，才能提高压缩率. 在实验中，基于 C-Store 决策树的压缩策略选择方法为整个列得到的压缩方法为位图编码，而本文提出的策略选择方法采用局部学习获得的压缩方案包括位图编码、游程编码和字典编码. 绝大部分的区的压缩方案为字典编码，一少部分区的压缩方案为位图编码和游程编码. 表 2 显示的是一个 item 在一部分压缩方案压缩之后的表示形式以及压缩之后的占用的空间的大小. 其中 X 为 0 或者 1，N 表示参与位图编码的不同值的数目. 例如 *customer* 表的 *mktsegment* 列中共有 5 个不同的值，采用位图编码的话，那么这里的 N 就为 5. 显然，采用字典编码更节省空间(位图编码还有另外一种实现方式，用一个位来表示压缩之后的数据，这样有一个缺点，很难定位一个item，不利于查询，如果再用一条记录来标记位置的话，占用的空间则更大).

表 2　压缩方法和压缩效果

压缩方法	压缩之后的结构	压缩之后占用的空间
字典编码	(字典编号)	1 字节
位图编码	(XXXXXXXX)	N 个字节
游程编码	(*value*,*begin_id*,*len*)	*Length*(*value*)+8+8

基于学习的压缩策略选择方法和基于统计分析的压缩策略选择方法的压缩效果相差不大. 但是随着数据量的增大，前者的压缩效果要稍优于后者. 这是因为基于学习的压缩策略选择方法在每一个区上都实现了最大限度的压缩，而基于统计分析的压缩策略选择方法则是通过随机抽取一定量的区的统计信息来得到整个列的统计信息，从而得到推荐压缩方法，这也就意味着这种方法在采样随机性的影响，因此，无法保证对每一个区上的推荐压缩算法是适当的压缩算法，造成了压缩比稍低于前者.

b) 压缩时间的比较. 主要是指对同样大小的数据，完成压缩所需要的时间. 如图 5 所示.

从图 5 可以看出，在数据量比较小的时候，3 种方法所用的压缩时间相差不大，而当数据量增大的时候，基于 C-Store 决策树的压缩策略选择方法需要的时间要稍长于本文提出的两种压缩策略选择方法. 这是因为，基于 C-Store 决策树的压缩策略选择方法需要对整个列的数据的全部统计信息进行统计，而本文提出的两种区级压缩模式下的压缩策略选择方法则不同，在大部分区中只需要统计部分的统计信息来验证本区适合采用推荐的压缩算法，只有在不满足推荐压缩算法的区才统计全部的统计信息，这样一来，随着数据量的增大，这种差别就变得比较明显.

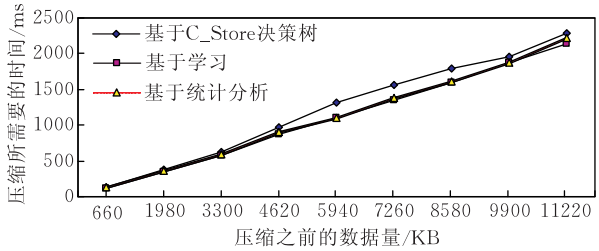

图 5　压缩所需时间图

从图 5 中可以看出，本文提出的基于学习的压缩策略选择方法和基于统计分析的压缩策略选择方法在压缩相同大小的数据时需要的压缩时间相差很小，偶尔后者需要的压缩时间要稍大于前者，这是因为后者采用的是通过随机抽取一定数量区的数据作为样本来估计整个列的数据分布，从而推测出最适合这个列的压缩方法，但是有时会发生估算错误，也就意味着样本数据的分布特点和这个列的数据分布特点不相符，这就意味着更多的区需要学习，也就会造成时间的浪费.

6　结　论

通过第 5 节的分析可以看出，本文提出的两种区级压缩模式下的压缩策略选择方法要优于基于 C-Store 决策树的压缩策略选择方法. 本文的两种区级压缩模式下的策略选择方法通过对局部数据的学习来推断后面的数据的分布，并通过对下一部分的数据的统计信息进行分析来验证是否符合推荐压缩方式的特点，从而对压缩算法进行校正，找到适合本部分的压缩方法，这也就意味着，每一个部分都得到了较好的压缩，从而表现出优于基于 C-Store 决策树的压缩策略选择方法的特点.

通过压缩比和压缩时间的比较，本文提出的基于学习的压缩策略选择方法要优于基于统计分析的压缩策略选择方法.

虽然在压缩比和压缩时间上表现出比 C-Store 决策树更优的性能,但是区的大小也是一个影响压缩效果的重要因素.区过大或者过小多是不利于压缩的.因此需要更进一步的实验来确定区的大小,并对数据仓库最重要的两类查询:范围查询和聚集查询进行测试.另外本文的算法中使用到了一些阈值,这些阈值在下一步的工作中进行测试确定.

参 考 文 献

[1] Stratos Idreos et al. Self-organizing tuple reconstruction in column-stores//Proceedings of the SIGMOD. Providence, Rhode Island, USA, 2009: 297-308

[2] Huffman D. A method for the construction of minimum-redundancy codes. IEEE Transactions on Information Theory, 1952, 9(40): 1098-1101

[3] Witten I H, Neal R, Cleary J. Arithmetic coding for data compression. Communications of the ACM, 1987, 30(6): 520-540

[4] Roth M A, Van Horn S J. Database compression. ACM SIGMOD Record, 1993, 22(3): 31-39

[5] Tanaka H, Leon-Garcia A. Efficient run-length encodings. IEEE Transactions on Information Theory, 1982, 6(28): 880-890

[6] Ziv J, Lempl A. A universal algorithm for sequential data compression. Proceedings of the IEEE Transactions on Information Theory, 1977, 22(1): 337-343

[7] Abadi D J et al. Query execution in column-oriented database systems [Ph. D. dissertation]. Cambridge, Massachusetts: Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2008

[8] Trondheim, Norway, Mike Stonebraker, Abadi D J et al. C-store—A column oriented DBMS//Proceedings of the 31st VLDB Conference. Trondheim, Norway, 2005: 553-564

[9] Weyla S, Friesb J, Wiederholdc G, Germano F. A modular self-describing clinical databank system. Computers and Biomedical Research, 1975, 8(3): 279-293

[10] Wong H K T et al. Bit transposed files//Proceedings of the 11th International Conference on Very Large Data Bases Stockholm. Sweden, 1985: 448-457

[11] Copeland G P et al. A decomposition storage model//Proceedings of the International Conference on Management of Data. Austin, Texas, United States,1985: 268-279

[12] Ailamaki A. A storage model to bridge the processor memory speed gap//Proceedings of the High Performance Transaction Systems Workshop (HPTS). Asilomar, CA, 2001: 131-136

[13] Ramamurthy R et al. A case for fractured mirrors//Proceedings of the 28th VLDB Conference. NJ USA, 2002: 89-101

[14] Poess M, Potapo D. Data compression in Oracle//Proceedings of the 29th International Conference on Very Large Data Bases. Berlin, Germany, 2003: 937-947

[15] Wong H K T et al. Bit transposition for very large scientific and statistical databases. Algorithmica, 1986, 1(3): 289-309

[16] Carsten Binnig Stefan Hildenbrand Franz Färber. Dictionary-based order-preserving string compression for main memory column stores//Proceedings of the SIGMOD. Providence, Rhode Island, USA, 2009: 283-296

[17] Zhao Kai. The research on compression and query processing methods of scientific and statistical databases [Ph. D. dissertation]. School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 2006(in Chinese)
(赵锴. 科学与统计数据库压缩与查询处理方法的研究[博士学位论文]. 哈尔滨工业大学计算机科学与技术学院, 哈尔滨, 2006)

WANG Zhen-Xi, born in 1985, master candidate. His main research interests include column storage, data compression.

LE Jia-Jin, born in 1951, professor, Ph. D. supervisor. His research interests include database and data warehouse, software engineering theory and practice.

WANG Mei, born in 1980, Ph. D.. Her primary research interests include database, data warehouse management, image semantic analysis, information retrieval.

LIU Guo-Hua, born in 1966, professor, Ph. D. supervisor. His main research interests include privacy, document copy detection, *k*-anonymity privacy protection model for the uncertainty of data management, data-oriented business process management.

Background

This work is supported by the National Natural Science Foundation of China (No. 60773100). The purpose of this project is to achieve a complete, column-oriented data warehouse management system. Such system is researched by nobody around the world and is a blank field. As the attributes of big mount of data and column-oriented, Compression technology becomes hot as the demand of query. The main research of column-oriented compression is focused on improving of compression method, but few is focused on how to chooses such method. This article proposes a method of compression which is based on extents and then a method of choosing strategy, improving the efficiency of compression, reducing the time of compression.