Journal of Computer Applications
计算机应用,2016,36(1):21 -26,51
ISSN 1001-9081
CODEN JYIIDU
2016-01-10
http://www.joca.cn

文章编号:1001-9081(2016)01-0021-06
DOI:10.11772/j.issn.1001-9081.2016.01.0021

压缩数据上的关系代数操作算法

丁鑫哲[1],张兆功[1*],李建中[2],谭　龙[1],刘　勇[1]
(1.黑龙江大学 计算机科学技术学院,哈尔滨 150080;　2.哈尔滨工业大学 计算机科学技术学院,哈尔滨 150010)
(*通信作者电子邮箱 zhaogong@mtu.edu)

摘　要:针对在大数据管理中,在压缩的数据上无需解压即可进行相关操作的问题,在数据服从正态分布的前提下,根据列数据存储的特点,提出了一种新的面向列存储的压缩方法——CCA。首先,通过对列数据的长度进行归类;然后,采用抽样的方法获得重复度较高的前缀;最后,使用字典编码进行压缩,提出了列索引(CI)和列实体(CR)作为数据压缩结构来降低大数据存储的空间需求,从而直接有效地在压缩数据上支持选择、投影、连接等基本操作,并实现了基于CCA的数据库原型系统——D-DBMS。理论分析和在1 TB数据上的实验结果表明,该压缩算法能够显著提高大数据的存储效率和数据操作性能,与BAP和TIDC压缩方法相比,在压缩率分别提高了51%、14%;在执行速度上提高了47%、42%。

关键词:大数据压缩;列索引;列实体;关系代数操作
中图分类号:TP311.13　　**文献标志码**:A

Relational algebraic operation algorithm on compressed data

DING Xinzhe[1], ZHANG Zhaogong[1*], LI Jianzhong[2], TAN Long[1], LIU Yong[1]
(1. *College of Computer Science and Technology, Heilongjiang University, Harbin Heilongjiang* 150080, *China*;
2. *College of Computer Science and Technology, Harbin Institute of Technology, Harbin Heilongjiang* 150010, *China*)

Abstract: Since in the massive data management, the compressed data can be done some operations without decompressing first, under the condition of normal distribution, according to features of column data storage, a new compression algorithm which oriented column storage, called CCA (Column Compression Algorithm), was proposed. Firstly, the length of data was classified; secondly, the sampling method was used to get more repetitive prefix; finally the dictionary coding was utilized to compress, meanwhile the Column Index (CI) and Column Reality (CR) were acted as data compression structure to reduce storage requirement of massive data storage, thus the basic relational algebraic operations such as select, project and join were directly and effectively supported. A prototype database system based on CCA, called D-DBMS (Ding-Database Management System), was implemented. The theoretical analyses and the results of experiment on 1 TB data show that the proposed compression algorithm can significantly improve the performance of massive data storage efficiency and data manipulation. Compared to BAP (Bit Address Physical) and TIDC (TupleID Center) method, the compression rate of CCA was improved by 51% and 14%, and its running speed was improved by 47% and 42%.

Key words: massive data compression; Column Index (CI); Column Reality (CR); relational algebraic operation

0　引言

在当今信息爆炸的时代里,每天都在产生着巨量的数据,如此巨量的数据如何存储、存储后如何进行高效的分析,这是一个亟待解决的问题。磁盘存储器单位容量价格的下降使得人们能够用更低的成本存储更多的数据,但对于大数据来说仍然无法满足需求。于是就出现了很多的云存储解决方案,但对于数据体量巨大的数据,在综合考虑安全性、实时性、传输性等方面的因素后,仍然未能得到广泛、全面的应用。

对于大数据的处理,一般情况下需要对数据进行过滤分析,并执行相应的操作以得到具体结果,例如对数据进行统计分析、机器学习、数据挖掘等,将现有的数据通过适当的算法进行筛选处理,最终得到相应的结果,其中相应的最基本操作即为代数操作。在大数据上进行一系列基础且高效的代数操作,目前有两种解决方法:一种是采用并行的方式,如采用Hadoop[1]解决;另一种是采用压缩技术。本文主要对大数据上的数据压缩技术进行讨论研究。对于压缩后的数据,通常需要整个数据完全解压后再进行计算,这也是采用压缩技术存在的瓶颈之一。如何在压缩的数据上进行无需解压的相关操作,获得更好的压缩效率和压缩比等,都是当前大数据压缩处理技术中存在的挑战。

本文在数据服从正态分布的前提下,针对列存储数据的特点,根据数据value的长度,提出了列索引(Column Index, CI)和列实体(Column Reality, CR)压缩结构及数据压缩算

收稿日期:2015-07-27;**修回日期**:2015-08-04。
基金项目:国家自然科学基金资助项目(81273649);黑龙江省自然科学基金资助项目(F201434)。
作者简介:丁鑫哲(1990-),男,福建泉州人,硕士研究生,主要研究方向:大数据压缩;　张兆功(1963-),男,黑龙江哈尔滨人,教授,博士,CCF会员,主要研究方向:海量数据挖掘、生物信息学;　李建中(1950-),男,黑龙江哈尔滨人,教授,CCF会员,主要研究方向:海量数据管理与计算、无线传感器网络;　谭龙(1971-),男,黑龙江哈尔滨人,副教授,博士,CCF会员,主要研究方向:无线认知网络、数据挖掘;　刘勇(1990-),男,黑龙江七台河人,硕士研究生,主要研究方向:关键字检索。

法——CCA(Column Compression Algorithm),分析了压缩效率,并实现了基于 CCA 的数据库原型系统——D-DBMS(Ding-DataBase Management System)。通过理论分析和实验,与相关无解压代数操作算法进行了对比,实验结果表明 CCA 在压缩率和无解压关系代数操作效率上优于目前相关的无解压压缩算法。

1 相关工作

文献[2]提出了 BTF(Bit Transposed File)的文件结构,在科学与统计数据库(Statistical and Scientific DataBase, SSDB)上采用 RLE(Run Length Encoding)方法对 Bit partitions 进行压缩,然后在压缩的 Bit vector 上进行查询操作,但只考虑了单表的查询操作,没有给出连接操作。文献[3]提出了 BAP(Bit Address Physical)压缩算法,能够在 SSDB 上进行快速双向映射操作,比传统的位图法、RLE、Header 压缩方法效果更好,但对于常量的比例有一定的要求。文献[4]提出了在压缩数据仓库上的无解压聚集算法,给出了线性化和反线性化函数,能够实现联机分析处理(OnLine Analytical Processing, OLAP),但未给出其他算法。文献[5]提出了一种在压缩数据上进行 Cube 计算的算法,在压缩数据上实现了完全映射。文献[6]提出了一种基于属性划分的海量数据高频度关系数据压缩存储方法 TIDC(TupleID Center),实现了选择、投影、连接操作,但要求常量频度要在 60% 以上才能有较好的效果。文献[7]提出了一种区级压缩模式,通过学习参照信息与当前区之间的相似性和差异性进行相关的压缩策略推荐方法,从而提高了压缩性能。文献[8]对现有的字典压缩方法进行了总结;同时提出新的压缩方法,并与其他算法进行了比较,具有较好的效果。文献[9]提出了在 ERP(Enterprise Resource Planning)的多属性数据上采用 CGK(Composite Group Key)的方式来进行压缩索引。文献[10]提出了基于列存储的数据库管理系统 C-Store,它通过建立决策树来判定每一列应该采用何种压缩算法。文献[11]对列存储数据的关键技术进行了相应的描述和总结。

综上所述,目前数据处理技术能够到达属性级别的无解压计算,但仍然需要进行 Backward Mapping,故会增加查询处理时间。目前在压缩数据上实现相关代数操作算法的有文献[6],但是其压缩算法对于常量比例的要求比较高。本文提出的 CCA 在数据服从正态分布的条件下,主要采用对数据 value 的长度进行分类处理,生成 CI 和 CR 压缩结构,从而达到较好的压缩效果和查询效率。

2 预备知识

本章描述相关的压缩数据上无解压计算的压缩算法,以及下文中所用的符号定义。

常见的压缩算法有 Run Length Encoding、Header Compression、BitMap、Dictionary Encoding、Golomb Coding、Huffman Coding 等。

目前在数据库上常见的压缩算法有游程编码、头部压缩、位图编码、字典编码、哈弗曼编码、哥伦布编码等方法。数据库的压缩一般在以下几个粒度中进行:表、块、元组、属性、比特[7]。

下面简单介绍一下 BAP、TIDC、Composite Group Keys 等压缩、索引查找算法。

2.1 BAP 压缩算法

文献[3]提出了一种新的针对 SSDB 数据去重的压缩编码方法。BAP 将压缩去重的数据称为常量,其他数据叫作非常量。压缩后的数据生成 3 个向量:***BV***(Bit Vector)、***AV***(Address Vector)、***PV***(Physical Vector)。

假设未压缩的数据为 $D = (d_1, d_2, \cdots, d_n)$。

1)***BV***。用来表示非常量在数据中出现的地方,非常量出现的位置用 1 表示,常量出现的位置用 0 表示。常量设为 C。

2)***AV***。用来表示非常量的个数与对应位置信息。

3)***PV***。用来记录非常量。

BV、***AV***、***PV*** 的形式化定义如下:

$\boldsymbol{BV} = (b_1, b_2, \cdots, b_i, \cdots b_n)$ if $b_i = C$ then $b_i = 0$ else $b_i = 1$。

$\boldsymbol{AV} = (a_1, a_2, \cdots, a_e, \cdots, a_m)$,$a_1 = 0$,$m$ 为划分数,a_e 表示第 e 分区前的非常量个数。

$\boldsymbol{PV} = (p_1, p_2, \cdots, p_j, \cdots, p_k)$,$1 \leqslant k \leqslant n$,$p_j \neq c$。BAP 压缩算法中有两个映射函数:Forward Mapping 和 Backward Mapping。第 1 个映射函数为给定 ***AV***,求 ***PV***;第 2 个映射函数是给定 ***PV***,求 ***AV***。

算法 1 BAP_Compression。

输入:原始数据 D。

输出:压缩后的数据 D'。

```
1)  预处理获得压缩长度 k,常量 C,常量大小 csize = 0
2)  For each 子部分
3)    For each d_i
4)      If(d_i == C) csize ++
5)      Else
6)        d_i 写入 PV 缓冲区;同时用 Golomb 编码,根据 k 和
          csize 的大小关系,处理不同的数据,然后写入
          BV 缓冲区和 AV 缓冲区
7)  将 BV、AV、PV 写入磁盘,得到 D'
```

BAP 压缩算法时间复杂度为 $O(2n) = O(n)$。

定义 1 压缩比。压缩比定义如下:

压缩比 = 原始数据大小 / 压缩后数据大小

假设 D 有 n 个元素,每个元素所占磁盘大小为 B_{PV},压缩后 ***AV*** 所占磁盘大小 B_{AV},子部分大小 S,磁盘块大小 B,常量比 cf,则 BAP 压缩算法的压缩比为:

$$\frac{n \times B_{PV}}{n/S \times B + n/S \times B_{AV} + n \times (1 - cf) \times B_{PV}}$$

2.2 TIDC

文献[5]提出的 TIDC 是基于划分的面向列存储的适合于海量高频度关系数据的压缩存储方法。

定义 2 高频度数据。对经过属性划分后的某列数据,存在某一数值 C,C 的频度大于 60%,则称此数据为高频度数据,称为常量。

TIDC 将整列数据按照磁盘块大小划分为不均匀的子部分,然后使用非常量的 TupleID 数值即 *TupleID-Value*(*TIDV*),与每部分的数据数量 *TupleID-Num*(*TIDN*) 来表示压缩后的数据。*TIDV*、*TIDN* 的形式化定义如下:

设 $D = (d_1, d_2, \cdots, d_n)$ 为原始高频数据,C 是常量

$TIDV = ((TID_1, Value_1), (TID_2, Value_2), \cdots, (TID_m, Value_m)); 1 \leqslant m \leqslant n$

$TIDN = (Num_1, Num_2, \cdots, Num_j, \cdots, Num_k); 1 \leqslant Num_j \leqslant n$

算法 2　TIDC_Compression。

输入:原始数据 D,常量 C。

输出:压缩后的数据 D'。

1)　元组数为 n,每个子部分大小为 m

2)　For each 子部分

3)　　For each d_i

4)　　If (d_i != C)

5)　　存 d_i 和其对应 *TupleID* 到 *TIDV* 缓冲区,若缓冲区满达到 m 则写入磁盘

6)　If (*TIDV* 不空)

7)　　写 *TIDV* 内容到磁盘

8)　　写最末元素的 *TupleID* 到 *TIDN*

9)　　清空 *TIDV*

10)　写 *TIDN* 到磁盘

压缩前每个数据的大小为 b,*TIDV* 中 *TupleID* 和 *Value* 所占用空间分别为 b_{ID}、b,*TIDN* 中每个元素占用大小 b_{TN},常量 C 的频度为 f,磁盘块大小为 B,则 TIDC 压缩算法的压缩比为:

$$\frac{n \times b}{(n \times (1-f) \times (b_{ID} + b))/B) \times (B + b_{TN})}$$

TIDC 压缩算法时间复杂度为 $O(2n) = O(n)$。

文献[5]在 TIDC 压缩算法上实现了无解压的选择、投影、连接等关系代数操作。设 *TIDV* 被划分为每部分有 m 个元素的 *num* 个子部分,待投影元组集合大小为 T,待投影属性集合大小为 M,*TIDN* 中元素的个数与 m 数值中较大的值为 k,则关系代数操作的时间复杂度如表 1 所示。

表 1　TIDC 关系代数操作时间复杂度

操作	时间复杂度
选择	$O(num \times m \log m)$
投影	$O(k \log k + TM)$
连接	$O(M \log M)$

2.3　CGK

文献[9]在分析 SAP ERP 应用数据的基础上,提出了 CGK 的索引方法,直接在字典压缩的数据上建立高效的索引。该方法会生成两个数据结构,一个是 key-identifier list,用 K 表示;一个是 position list,用 P 表示。KI 中包含整数键值 k_{id}。最终通过使用改进的 B + Tree 来达到快速查找的目的。

索引的创建过程如下,如图 1(a)所示。图 1(a)中的编码对应图 1(b)中的字典。

1)所有需要组合在一起的值组合成 K 的向量 $\boldsymbol{K}_U$,在加上递增列表 *row-ids*(P_U);

2)然后根据 K 排序,得到最终的 K 和 P。

设数据表长为 n,组合 *key* 的长度为 k,查询的时间复杂度为 $O(k \log n)$。

2.4　符号定义

具体的符号定义见表 2。

3　CCA

3.1　压缩存储结构

CCA 读取列文件,主要用于查询,不进行插入、删除、更新等 CUD 操作。

定义 3　列文件。

面向列存储的数据库中逻辑行的数据内容为 $R = (r_1, r_2, \cdots, r_N)$,按每一个 r_i 为一个属性,按每个属性一个文件进行存储,成为列文件。设列文件存储格式为(*key*,*value*),*key* 是对应 *value* 值在列文件中的唯一值。在数据服从 $N(\mu,\sigma^2)$ 正态分布的条件下,列文件 *key* 值是从小到大递增的。

CCA 采用 CI 和 CR 来存储(*key*,*value*)压缩后的数据。

CI(Column Index),用来存储 *value* 的长度值与 *value* 长度值所在 CR 中的对应位置。形式化定义:$CI_i = (value\ length, index_{k_j}), 1 \leqslant i \leqslant k, 1 \leqslant j \leqslant n$。

CR(Column Reality),用来存储 *value* 及 *key*。形式定义:$CR_i = (value, index_{c_j}), 1 \leqslant i \leqslant p, 1 \leqslant j \leqslant n$。

CI 和 CR 结构示意如图 2(a)、(b)。

其中 $1 \leqslant k \leqslant p \leqslant n$,每个 $value\ length_k$ 的 $index_{k_j}$ 个数都不尽相同,每个 $value_p$ 中的 $index_{c_j}$ 亦是如此。

表 2　文中所用符号含义

符号	含义	符号	含义
B	磁盘块大小	k_j	CI 中每个长度里 *key* 的下标
n	数据元组数	$7L_i$	前缀长度
μ	*value* 平均长度	*DICT*	前缀字典
m	前缀个数	*SIZE_O*	原始数据大小
k	*value* 有 k 种长度	*SIZE_C*	压缩后数据大小
p	*value* 有 p 种不同的值	*SIZE_C'*	采用前缀后压缩数据大小
s	$s = p - m$	*SIZE_CI*	CI 大小
a_i	CR 中 *value* 相同长度的个数	*SIZE_CR*	CR 大小
b_i	CI 中每个 *value* 的长度	*SIZE_DICT*	前缀字典大小
c_i	CR 中个 *key* 的个数		

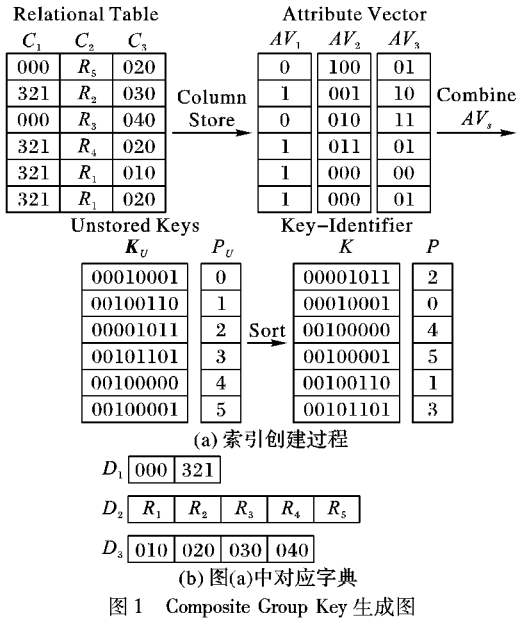


图 1　Composite Group Key 生成图

3.2　压缩存储方法

CCA 扫描数据一遍即可获得压缩文件。方法是扫描列数据的时候,读入每个元组(*key*,*value*),求出 *value* 的长度 b_i,到 *ci_buffer* 中查找是否长度 b_i 已经存在,不存在则在 *ci_buffer* 中创建一个,存在则在 *ci_buffer* 中写入其在 *cr_buffer* 中的位置;同时把 *key* 值写入 *cr_buffer* 中 *value* 所对应的位置。算法如

下。

value $length_1$	$index_1$	$index_2$	…	$index_{k_j}$
value $length_2$				
⋮				
value $length_k$				

(a) Column Index结构示意图

$value_1$	$index_1$	$index_2$	…	$index_{c_j}$
$value_2$				
⋮				
$value_p$				

(b) Column Reality结构示意图

图 2 CI 和 CR 的结构示意图

算法 3 CCA_Compression_1。

输入:原始数据 D。

输出:压缩后的数据文件 CI 和 CR。

1) For each 列文件
2) 读取(*key*,*value*),*value_length* = *len*(*value*)
3) If (*value_length* 在 *ci_buffer* 中)
4) If (*value* 在 *cr_buffer* 中)
5) 把 *key* 写到 *cr_buffer* 对应 *value* 的位置
6) Else
7) 在 *cr_buffer* 中创建 *value* 并保存 *key*
8) Else
9) 在 *ci_buffer* 中创建 *value_length*
10) 在 *cr_buffer* 中创建 *value* 并保存 *key*
11) If (*buffer* 满)
12) 把*buffer* 对应内容写入文件;同时将 *buffer* 中除了第一列之外的内容清空

不断地将 *ci_buffer* 和 *cr_buffer* 的内容写入磁盘,最终得到的就是压缩后的 CI 和 CR 文件。

举例说明如下:

D_{key} = (1,2,3,4,5,6,7,8,9,10)

D_{value} = (word,hello,work,work,hello,word,hello,hello,hello,hello)

CI 如下:(4,1,3),(5,2)

CR 如下:(word,1,6),(hello,2,5,7,8,9,10),(work,3,4)

当列数据具有一部分相同前缀时,加入字典编码的方法能够更进一步地提高压缩效率。首先对数据进行抽样,对于抽样到的数据,对数据每隔 4 个字符匹配一次前缀,对这些数据生成字典,使用 4 字节为其编码,总共支持 4 294 967 296 种编码,在大多数情况下已经足够使用。

采用抽样的 CCA 如下。

算法 4 CCA_Compression_2。

输入:原始数据 D。

输出:压缩后的数据文件 CI、CR 及前缀字典 DICT。

1) For each 列文件
2) 采用抽样的方法,获得 m 个不同的前缀,并采用字典编码的方式对其编码,得到 DICT
3) 读取(*key*,*value*),*value_length* = *len*(*value*)
4) If (*value_length* 在 *ci_buffer* 中)
5) If (*value* 在 *cr_buffer* 中)
6) 把 *key* 写到 *cr_buffer* 对应 *value* 的位置
7) Else
8) 在*cr_buffer* 中创建 *value*(*value* 的前缀采用编码进行替换) 并保存 *key*
9) Else
10) 在 *ci_buffer* 中创建 *value_length*
11) 在*cr_buffer* 中创建 *value*(*value* 的前缀采用编码进行替换) 并保存 *key*
12) If (*buffer* 满)
13) 把*buffer* 对应内容写入文件;同时将 *buffer* 中除了第 1 列之外的内容清空

3.3 压缩效率

读取列文件时,每个 *key* 占用的长度为实际长度,例如 *key* = 1 234 567,则占用 7 个字符,行数据库存储时,会预留每个 *key* 最长的宽度。*key* 存入 CR 时采用二进制方式写,故当数据量在 4 294 967 296 条记录内时,可以采用 4 字节的二进制整数来存储 *key* 值。当数据量大于 4 294 967 296 条时,可以根据需求适当拓展存储 *key* 的字节数。

数据服从 $N(\mu,\sigma^2)$ 正态,根据 CCA,原数据大小:

$$SIZE\_O = \sum_{i=1}^{n} b_i \times c_i$$

压缩后的结构 CI 和 CR 的大小:

$$SIZE\_CI = 4k + 4\sum_{i=1}^{k} a_i$$

$$SIZE\_CR = \sum_{i=1}^{p} (b_i + 4c_i)$$

压缩后的文件大小:

$$SIZE\_C = SIZE\_CI + SIZE\_CR = 4k + 4\sum_{i=1}^{k} a_i + \sum_{i=1}^{p} (b_i + 4c_i)$$

数据服从正态分布,所以有 $\sum_{i=1}^{p} b_i = k\mu$, $\sum_{i=1}^{p} c_i = n$, $\sum_{i=1}^{k} a_i = p$,所以上述式子可以化简如下:

$$SIZE\_O = 4n + n\mu$$

$$SIZE\_CI = 4k + 4p$$

$$SIZE\_CR = k\mu + 4n$$

$$SIZE\_C = 4k + 4p + k\mu + 4n$$

设 $y = SIZE\_O - SIZE\_C = 4n + n\mu - (4k + 4p + k\mu + 4n) > 0$ 即表明有压缩效果。经计算可得,当 $\mu > 4(k + p)/(n - k)$ 时 CCA 就有压缩效果。

根据求出的 *SIZE_O* 和 *SIZE_C* 就可以得到 CCA_Compression_1 的压缩比为:

$$\frac{SIZE\_O}{SIZE\_C} = \frac{4n + n\mu}{4k + 4p + k\mu + 4n}$$

3.4 无损性

定理 1 当列数据的值只有一种长度,且 100% 无重复时,$\mu > 4$ 时 CCA 就有压缩效果。

证明 即 $k = 1, p = n$ 的情况,$4(k + p)/(n - k) = 4(1 + n)/(n - 1)$,$n$ 为大数据上元组的个数,故 $n \gg 1$,故 $4(1 + n)/(n - 1) = 4$,即 $\mu > 4$ 时,$y > 0$,也就是 CCA 有压缩效果。 证毕。

对于 CCA_Compression_2,设经过抽样得到 m 个重复率较高的不同前缀,对其进行编码,$SIZE\_DICT = \sum_{i=1}^{m} L_i + 4m$,其中 $SIZE\_CR'$ 为应用字典编码后 CR 的大小,则压缩后总大小为:

$$SIZE\_C' = SIZE\_CI + SIZE\_CR' + SIZE\_DICT = 4k +$$

$$4\sum_{i=1}^{k}a_i+\sum_{i=1}^{p}(b_i-L_i+4c_i)+\sum_{i=1}^{m}L_i+4m=4k+$$

$$4\sum_{i=1}^{k}a_i+\sum_{i=1}^{p}b_i+\sum_{i=1}^{p}4c_i-\sum_{i=1}^{p}L_i+\sum_{i=1}^{m}L_i+4m=$$

$$4k+4p+k\mu+4n+4m+\sum_{i=1}^{m}L_i-\sum_{i=1}^{p}L_i$$

若 $4m-\sum_{i=1}^{s}L_i\leqslant 0$,推出 $4m\leqslant\sum_{i=1}^{s}L_i$。

根据求出的 *SIZE_O* 和 *SIZE_C'* 就可以得到 CCA_Compression_2 的压缩比为:

$$\frac{SIZE\_O}{SIZE\_C'}=\frac{4n+n\mu}{4k+4p+n\mu+4n+\sum_{i=1}^{m}L_i-\sum_{i=1}^{p}L_i}$$

定理 2　当列数据的值 100% 无重复时,至少有 4 个字符的重复前缀,则 CCA 就有压缩效果。

证明　令 $L_i=4$,则有 $4m\leqslant 4(1+2+\cdots+s)$。

因为 $m+s=p=n$,由于采用抽样的方法,$n\gg m$,所以 $s\gg m$,所以 $4m\leqslant 4(1+2+\cdots+s)$ 成立。　证毕。

3.5　算法复杂度分析

算法 CCA_Compression_1 的时间复杂度为 $O(n\log p+n)$,其中定位到 *ci_buffer* 中对应位置需要 $O\left(\sum_{j=1}^{k}k_j\right)$,定位到 *cr_buffer* 中对应位置需要 $O\left(\log p+\sum_{j=1}^{p}c_j\right)$,最坏情况即为无重复 *value*,此时间复杂度为 $O(n\log(p+k)+n)$。

算法 CCA_Compression_2 的时间复杂度为 $O(m\log m+n\log p+n)$。

4　基于 CCA 的关系代数操作算法

4.1　选择操作

选择操作分为 3 种:1) 知 *key* 求 *value*;2) 知 *value* 求 *key*;3) 进行 *op* 操作,$op\in\{<,\leqslant,>,\geqslant\}$,*num* 为 *value* 存在的不同磁盘块个数。

算法 5　CCA_Select_ByKey(*key*)。

输入:*key*。

输出:*key* 对应的 *value*。

1)　直接定位到 CR 的第二列 *cr2*
2)　　在 *cr2* 上采用二分查找
3)　　If (第 2) 步找到 *key* 所在的范围 *f1*、*f2*)
4)　　If (到 *f1*、*f2* 中二分查找到 *key*)
5)　　Return (*key*,*value*)
6)　　Else Return None
7)　　Return None

时间复杂度为 $num*\log p\log c_j$。

算法 6　CCA_Select_ByValue(*value*)。

输入:*value*。

输出:*value* 对应的 *key*。

1)　获取 *value* 长度 *len_value*
2)　在 CI 中二分查找
3)　If (第 2) 步找到 *len_value* 在 CR 中的位置 *position*)
4)　　Return *position* 中的(*key*,*value*)
5)　Return None

时间复杂度为 $num*\log k$。

算法 7　CCA_Select_ByOpValue(*op*,*value*)。

输入:相关操作符 *op*,*value*。

输出:满足条件的元组集合 *result*。

1)　获取 *value* 长度 *len_value*
2)　For each *Key*,*Value* 所在磁盘块
3)　　If *Value* 满足 *op value*
4)　　*result* = *result* ∪ (*key*,*value*)
5)　Return *result*

时间复杂度为 $num*\log k\log p$。

4.2　投影操作

算法 8　CCA_Project($column_{attribute}$,$column_{key}$)。

输入:待投影属性集合 $column_{attribute}$,待投影元组号集合 $column_{key}$。

输出:投影结果集合 *result*。

1)　For each 属性 $A\in column_{attribute}$
2)　　For each 元组号 $K\in column_{key}$
3)　　*result* ∪ = 在 *A* 上作满足 *K* 的投影
4)　Return *result*

时间复杂度为 $num\_of\_A*c_j\log k$。

4.3　连接操作

为了考察压缩算法的存储性能,本文只实现了在两个压缩列上作等值连接操作。

算法 9　CCA_Join(A_i,A_t)。

输入:待连接的两列压缩文件 *AI*、*AT*,大小为 *I*、*T*,对应连接条件集合 A_i、A_t。

输出:*result* = {$(a_i,a_t)\mid a_i\in A_i,a_t\in A_t$ 且 a_i、a_t 对应值相等}。

1)　Function getResult(*x*,*y*)
2)　　For each 元素 in *x*,*y*
3)　　If ($x.value_i$ == $y.value_t$)
4)　　{(a_i,a_t) 加入 *result*}
5)　If (A_i,A_t 为 *key*)
6)　　For each A_i,A_t
7)　　直接定位到 CR 的第 2 列 *cr2*
8)　　在 *cr2* 上采用二分查找,return (*key*,*value*)
9)　　得到 A_i,A_t 的 *key*,*value* 元组集合 *r1*,*r2*
10)　　Return getResult(*r1*,*r2*)
11)　Else
12)　　For each A_i,A_t
13)　　获取 *value* 长度 *len_value*
14)　　在 CI 中二分查找
15)　　If (第 14) 步找到 *len_value* 在 CR 中的位置 *position*)
16)　　得到 A_i,A_t 的 *key*,*value* 元组集合 *r1*,*r2*
17)　　Return getResult(*r1*,*r2*)

时间复杂度如下:MAX($num_i*\log p\log c_{ij}$,$num_t*\log p\log c_{tj}$,$Q\log Q(Q=\mathrm{MAX}(I,T))$)。

5　实验

本文实现了一个数据库原型系统——D-DBMS,在这个原型系统上实现了 BAP、TIDC 和 CCA 压缩方法,目前系统能对列数据文件进行批量载入压缩存储,以及选择、投影、连接等关系代数操作。

5.1　实验设置

实验环境:Intel Xeon E3,3.30 GHz 四核,内存 16 GB,硬盘 2 TB,操作系统 Windows 7。所有算法和数据库原型系统全部采用 Python 2.7.6 编写。实验数据采用部分真实学生数据 148 GB信息 STU(Student Data Set)及采用 TPC-H(H version of Transaction Processing Performance Council)[12] 生成 858 GB 信

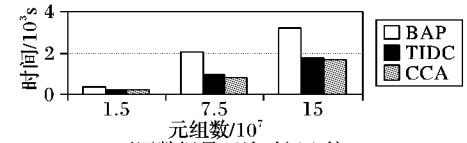
息 TPCHD(TPC-H Data Set);同时按照一定规则生成 STU 与 TPCHD 之间的关联信息 22 GB,数据量 1 028 GB。

5.2 压缩比的比较

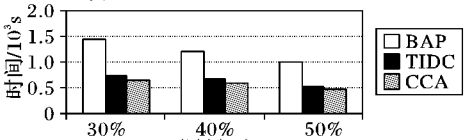
实验 1 对 BAP、TIDC、CCA 压缩时间进行比较。考察压缩时间与数据量、常量频度类型进行比较。

实验 2 对 BAP、TIDC、CCA 压缩比进行比较。考察压缩比常量频度、数据分布类型进行比较。

从图 3 可看出,CCA 具有较快的压缩效率。BAP 需对数据先进行 0,1 编码,然后在进行 Golomb 编码,故速度要比 TIDC 慢。TDIC 与 CCA 压缩时间接近,因为 TIDC 对数据进行两遍扫描,而 CCA 只对数据进行一遍扫描的同时作相关压缩操作。二者在常量频度相同时时间复杂度相近。



(a) 不同数据量压缩时间比较



(b) 不同常量频度压缩时间比较

图 3 不同数据量、常量频度压缩时间比较

由图 4 可以看出 CCA 相比 BAP 和 TIDC 有较好的压缩率,因为 BAP 和 TIDC 都要求常量的比例要在较高(大于 60%)的时候才具有较好的压缩率,CCA 有点类似于采用了多个常量的方法,所以压缩效率会更好。

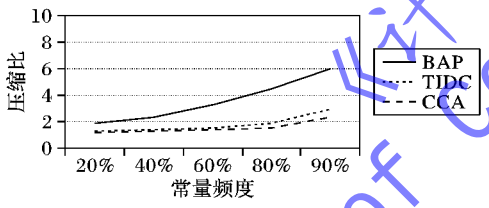


图 4 不同常量频度的压缩比

5.3 压缩数据上的不同操作算法效率对比

实验 3 CCA 与 BAP、TIDC 作单表选择操作时间比较,D-DBMS 目前实现 and 连接操作。元组数量为 21 亿,属性数量为 5,常量频度从 30% 到 90%。

实验 4 CCA 与 BAP、TIDC 作单表投影操作时间比较。元组数量为 21 亿,属性数量为 5,常量频度从 30% 到 90%。

实验 5 CCA 与 BAP、TIDC 作连接操作时间比较,D-DBMS目前实现了双表等值连接操作。两个元组数量分别为1 000万和 5 000 万,属性数量为 5,常量频度从 30% 到 90%。

图 5 反映的是常量频度对选择操作的影响,此处由于 BAP 需要进行解压操作,且需根据 BV 定位后再找 PV,开销较大,故在这里不作比较。TIDC 需要扫描一遍数据,CCA 对于一些选择操作可以直接定位,故比 TIDC 效率高。

图 6 显示的是常量频度对投影的影响,BAP 需要解压后计算;同时磁盘 I/O 较大,CCA 对于投影查询的数据都保存在相近的位置,故效率会比 TIDC 高一些。图 7 由于 BAP 作连接操作时间复杂度太高,故不作比较。当常量频度在 90% 时两者连接操作时间相近,但当频度小于 90% 时,CCA 具有更好的效果,因为 CCA 对于相近的数据能够直接获取。

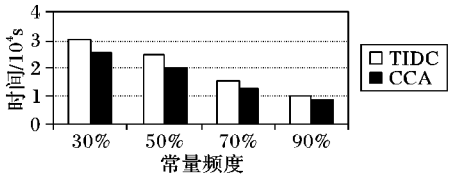


图 5 常量频度对选择操作的影响

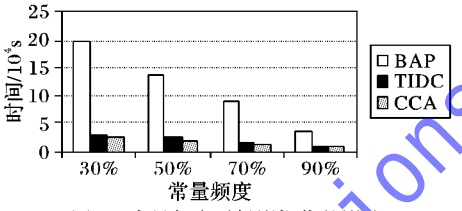


图 6 常量频度对投影操作的影响

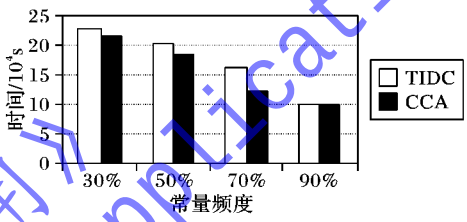


图 7 常量频度对连接操作的影响

6 结语

本文提出了 CCA,在数据服从正态分布的条件下,根据面向列存储的数据特点,采用针对 value 的长度的分类方法,并对 key 值进行压缩;同时使用抽样的方式获得多个重复率较高的前缀,并对其进行编码,从而得到更高的压缩率。本文实现了一个数据库原型系统——D-DBMS,并采用 CCA 实现了数据库上无解压的选择、投影、连接等关系代数操作,并与相关无解压计算算法进行了对比,与 BAP(Bit Address Physical)和 TIDC(TupleID Center)压缩方法相比,在压缩率分别提高了 51%、14%,在执行速度上提高了 47%、42%。理论分析和实验结果表明 CCA 是一个大的压缩数据上的高效无解压计算算法。

参考文献:

[1] LIN Y, AGRAWAL D, CHEN C, et al. Llama: leveraging columnar storage for scalable join processing in the MapReduce framework [C]// Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. New York: ACM, 2011: 961 - 972.

[2] WONG H K T, LI J, OLKENG F, et al. Bit transposition for very large scientific and statistical databases [J]. Algorithmica, 1986, 1 (1): 289 - 309.

[3] LI J, ROTEM D, WONG H K T. A new compression method with fast searching on large database [C]// Proceedings of the 13th International Conference on Very Large Data Bases. San Francisco, CA: Morgan Kaufmann, 1987: 311 - 318.

[4] LI J, SRIVASTAVA J. Efficient aggregation algorithms for compressed data warehouses [J]. IEEE transactions on knowledge and data engineering, 2002, 14(3): 515 - 529.

[5] WU W, GAO H, LI J. New algorithm for computing cube on very large compressed data sets [J]. IEEE transactions on knowledge and engineering, 2006, 18(12): 1667 - 1680.

(下转第 51 页)

[6] PHITHAKKITNUKOON S, HORANONT T, DI LORENZO G, et al. Activity-aware map: identifying human daily activity pattern using mobile phone data [C]// HBU'10: Proceedings of the First International Conference on Human Behavior Understanding. Berlin: Springer, 2010: 14 – 25.

[7] ISAACMAN S, BECKER R, CÁCERES R, et al. Identifying important places in people's lives from cellular network data [C]// Proceedings of the 9th International Conference on Pervasive Computing. Berlin: Springer, 2011: 133 – 151.

[8] TRAAG V A, BROWET A, CALABRESE F, et al. Social event detection in massive mobile phone data using probabilistic location inference [C]// Proceedings of the 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom). Piscataway, NJ: IEEE, 2011: 625 – 628.

[9] QUERCIA D, LATHIA N, CALABRESE F, et al. Recommending social events from mobile phone location data [C]// ICDM 2010: Proceedings of the 2010 IEEE 10th International Conference on Data Mining. Piscataway, NJ: IEEE, 2010: 971 – 976.

[10] CALABRESE F, COLONNA M, LOVISOLO P, et al. Real-time urban monitoring using cell phones: a case study in Rome [J]. IEEE transactions on intelligent transportation systems, 2011, 12(1): 141 – 151.

[11] SOTO V, FRIAS-MARTINEZ V, VIRSEDA J, et al. Prediction of socioeconomic levels using cell phone records [C]// Proceedings of the 19th International Conference on User Modeling, Adaption and Personalization. Berlin: Springer, 2011: 377 – 388.

[12] GIRARDIN F, VACCARI A, GERBER A, et al. Quantifying urban attractiveness from the distribution and density of digital footprints [J]. International journal of spatial data infrastructures research, 2009, 4: 175 – 200.

[13] GONZALEZ M C, HIDALGO C A, BARABASI A L. Understanding individual human mobility patterns [J]. Nature, 2008, 453(7196): 779 – 782.

[14] GIANNOTTI F, NANNI M, PEDRESCHI D, et al. Unveiling the complexity of human mobility by querying and mining massive trajectory data [J]. The VLDB journal, 2011, 20(5): 695 – 719.

[15] SIMINI F, GONZÁLEZ M C, MARITAN A, et al. A universal model for mobility and migration patterns [J]. Nature, 2012, 484(7392): 96 – 100.

[16] SONG C, QU Z, BLUMM N, et al. Limits of predictability in human mobility [J]. Science, 2010, 327(5968): 1018 – 1021.

Background

This work is partially supported by the National Basic Research Program (973 Program) of China (2012CB316203), the National Natural Science Foundation of China (61170085, 61472141, 61370101).

KONG Yangxin, born in 1991, M. S. candidate. His research interests include technology and application of location-based services, data mining.

JIN Cheqing, born in 1977, Ph. D., professor. His research interests include data stream management, location-based services, management of uncertain data.

WANG Xiaoling, born in 1975, Ph. D., professor. Her research interests include data-intensive computing management, technology and application of location-based services.

(上接第 26 页)

[6] 贾均刚, 张炜, 高宏. TIDC: 一种基于属性划分的高频度关系数据压缩存储方法[C]//第二十五届中国数据库学术会议(NDBC2008)论文集. 桂林: [出版者不详], 2008: 14 – 22. (JIA J G, ZHANG W, GAO H. TIDC: one kind based on attribute division high frequency relations data compression memory method [C]// Proceedings of the 25th National Database Conference. Guilin: [s. n.], 2008: 14 – 22.)

[7] 王振玺, 乐嘉锦, 王梅, 等. 列存储数据区级压缩模式与压缩策略选择方法[J]. 计算机学报, 2010, 33(8): 1523 – 1530. (WANG Z X, LE J J, WANG M, et al. Row stored datum area level compact model and compression strategy choice method [J]. Chinese journal of computers, 2010, 33(8): 1523 – 1530.)

[8] MÜLLER I, RATSCH C, FRBER F. Adaptive string dictionary compression in in-memory column-store database systems [C]// Proceedings of the 17th International Conference on Extending Database Technology. Athens: [s. n.], 2014: 152 – 158.

[9] FAUST M, SCHWALB D, PLATTNER H. Composite group-keys space-efficient indexing of multiple columns for compressed in-memory column stores [C]// IMDM 2013: Proceedings of the First and Second International Workshops on In-Memory Data Management and Analysis. Berlin: Springer, 2014: 42 – 54.

[10] STONEBRAKER M, ABADI D J, BATKIN A, et al. C-store: a column-oriented DBMS [C]// Proceedings of the 31st International Conference on Very Large Data Bases. [S. l.]: VLDB Endowment, 2005: 553 – 564.

[11] 李超, 张明博, 邢春晓, 等. 列存储数据库关键技术综述[J]. 计算机科学, 2010, 37(12): 1 – 7. (LI C. ZHANG M B, XING C X, et al. Survey and review on key technologies of column oriented database systems [J]. Computer science, 2010, 37(12): 1 – 7.)

[12] 康强强, 江舟, 金澈清, 等. TPCHSuite: 一个 TPC-H 自动化测试工具的设计与实现[J]. 计算机研究与发展, 2013, 50(z1): 394 – 398. (KANG Q Q, JIANG Z, JIN C Q, et al. TPCHSuite: the design and implementation of an automatic test tool for TPC-H [J]. Journal of computer research and development, 2013, 50(z1): 394 – 398.)

Background

This work is partially supported by the National Natural Science Foundation of China (81273649), the Natural Science Foundation of Heilongjiang Province (F201434).

DING Xinzhe, born in 1990, M. S. candidate. His research interests include big data compression.

ZHANG Zhaogong, born in 1963, Ph. D., professor. His research interests include massive data mining, bioinformatics.

LI Jianzhong, born in 1950, professor. His research interests include massive data management and computing, wireless sensor network.

TAN Long, born in 1971, Ph. D., associate professor. His research interests include wireless cognitive network, data mining.

LIU Yong, born in 1990, M. S. candidate. His research interests include keyword search.