一种融入 MD5 的影子表算法研究*

袁 满, 康峰峰, 黄 刚
(东北石油大学 计算机与信息技术学院, 黑龙江 大庆 163318)

摘 要: 为了快速提取源头数据、快速识别变化记录以及实现数据的快速增量提取,在剖析传统影子表法的工作原理上,提出基于 MD5 算法的影子表法的改进型线性算法,对对比表进行线性扫描,排除了不必要的回扫操作;同时通过 MD5 算法计算整条记录的"指纹",降低了字符串比对次数和时间,能够迅速识别出发生变化的记录。对所提出算法进行了应用测试,结果表明通过融入 MD5 算法后的影子表法提高了数据提取效率。基于影子表的增量提取方法是一种通用的增量捕获方法,能在任何数据库上实现;应用程序可以方便地在多种平台间移植,因此很适合解决异构数据库复制问题。

关键词: 影子表; 增量提取; 数据库复制; 线性
中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2013)01-0149-03
doi:10.3969/j.issn.1001-3695.2013.01.037

Research of MD5 shadow table method based on MD5 algorithm

YUAN Man, KANG Feng-feng, HUANG Gang
(*College of Computer & Information Technology*, *Northeast Petroleum University*, *Daqing Heilongjiang* 163318, *China*)

Abstract: As for the rapid extraction of source data, the rapid identification of changing data and the rapid incremental extraction of data, underlying the analysis of the working principle of the traditional shadow table, this paper proposed a kind of improved linear algorithm which merged MD5 algorithm into the ST. It scanned compared tables for the linear and eliminates unnecessary inverse operation. During scanning these tables, it employed MD5 algorithm to calculate the whole record 'fingerprint', it reduced the frequency and duration of string matching, and it could quickly recognize the changed records. The algorithm was verified practically, and the result shows that the proposed algorithm can improve the efficiency of data extraction in database. Incremental extraction method based on shadow table is a general incremental capturing method, and it can be implemented in any database. These applications can easily be transplanted in all kinds of platforms, so it is suitable to solve the heterogeneous database replication issues.

Key words: shadow table(ST); incremental extraction; database replication; linear

基于影子表法(ST)的增量提取方法在任何数据库中都能实现,因而常常用来解决异构数据库复制[1]。目前影子表法具体实现方式有两种:a)基于 NOT EXISTS 的 SQL 语句[2],其扫描策略需要至少进行 $M \times N$ 次比较,即时间复杂度为 $O(n^2)$,复杂程度过高;b)使用基于字符串匹配的方法[3],其主要是将每个元组的元素拼接成字符串,然后根据字符串匹配规则进行比较,如 BM 和 BMI 算法[4]。不同的字符串匹配算法虽然大大提高了相似记录的匹配速度,但是在浏览数据时原始数据需要与每一条新到记录集中的记录作比较,时间复杂度为 $O(m \times n)$,效率也较低。本文提出的 MD5 影子表法(MD5 shadow table,MD5-ST),通过 MD5 算法变换属性字段和线性比较数据库记录的方式,降低了算法的时间复杂度,极大地降低了比对次数,提高了比对效率。

1 融入 MD5 的影子表算法

1.1 算法概述

影子表法就是为当前数据库中的所有同步对象表 T 构建一个影子表 S,初始化时将源表中的信息同步到影子表中,也就是作一份当时的拷贝,以后就可在适当时机通过比较当前 T 和 S 的内容来获取净变化信息。基于影子表法能在任何数据库上实现,显示出它具有良好的适应性;独立于任何应用程序和支持平台,因此很适合解决异构数据库同步;影子表法的代价只有一倍的存储空间和不高的管理成本,由于得到的是净变化值,传输效率还能进一步提高,这些都是这种方法的优点[5,6]。

MD5[7~10](message-digest algorithm 5,信息摘要算法版本 5)在 20 世纪 90 年代初由 MIT Laboratory for Computer Science 和 RSA Data Security Inc 的 Ronald L. Rivest 开发出来,经 MD2、MD3 和 MD4 发展而来。MD5 将任意长度的信息流变换成一个 128 bit 的数,并且其是一个不可逆的加密算法。任何一个数据,无论是长数据或短数据,也不管是何种类型的数据,都有且只有一个独一无二的 MD5 值。只要数据本身发生变化,它的 MD5 值也会发生改变,发生碰撞的几率很小,也就是两个不同源数据的 MD5 值相同的情况微乎其微。因此,利用

收稿日期: 2012-04-16; **修回日期**: 2012-06-08 **基金项目**: 黑龙江省教育厅基金资助项目(11541008)

作者简介: 袁满(1965-),男,吉林东安人,教授,硕导,博士(后),CCF 会员,主要研究方向为信息集成、高级数据管理、数据库及应用;康峰峰(1987-),男,山西太原人,硕士研究生,主要研究方向为软件工程与集成技术(kffbest@163.com);黄刚(1980-),男,黑龙江哈尔滨人,硕士,主要研究方向为企业信息集成.

这一原理,可以事先在影子表 S 中增加一个字段来存放 MD5 值(MD5-ST),该表除了包括源表的完整备份外,还记录了根据所有字段的数据计算出来的 MD5 值。这样,在每次进行数据比对时,对源表和 MD5 影子表进行 MD5 值的比对,从而决定源表中的数据是新增、修改还是删除,同时更新影子表中的 MD5 值。通过 MD5 算法变换属性字段后的字符串来进行比对,避免了元组每个属性字段的匹配,降低了字符串匹配的次数和时间。

1.2 算法描述

该算法的过程可以分为三个阶段:

a)准备阶段。首先进行关键字的选择,关键字是由一个或者几个属性组成的集合,能够唯一地标志一条记录。一般来说,需要根据实际的应用背景去确定主键。将确定的主键存储到映射关系表 M 中;将需要进行增量数据提取的字段按照一定的秩序作连接运算得到一个新的字符串,将该字符串进行 MD5 运算得到的 MD5 值存储到影子表中。

b)排序阶段。通过确定的关键字对记录进行排序,得到对关键字有序的数据集合。

c)对比阶段。通过比较源表和影子表主键大小,确定记录作何种修改。当主键值相同时,计算源表 MD5 值并与存储在影子表中的 MD5 值进行比较,当 MD5 值相同时该记录没有作任何修改,当 MD5 值不同时该记录为修改记录;当主键值不同时,如果源表主键值大于影子表主键值,则影子表记录为要删除记录,如果源表主键值小于影子表主键值,则源表记录为新增记录。

本文用 T_Pointer,S_Pointer 代表分别指向两个待比较结果集的指针。运算时,通过读取映射表 M,对比源表 T 和影子表 S,获得的净变化值添加到增量表 C 中。MD5-ST 算法的伪代码如下:

```
{
Tkey = Getkey (M) ;
        Skey = Getkey (M) ; //获取对比表主键
        T[N] = Sort(Tkey) ;
        S[N] = Sort(Skey) ;/ * 根据取得的主键,分别将源表和影子表排序,得到两个待比较的结果集 * /
while(T. HasRows && S. HasRows)
        //源表和影子表子指针都没有移动到表尾
{
FindData(T_Pointer, S_Pointer) ;
//新增指针 T_Pointer, S_Pointer,分别指向两个数据集的记录
F = Compare(Tkey ,Skey) ;//比较指针所指记录的主键
/ * 开始判断增量数据是何种数据操作类型,用 1、2、3 分别标记 Insert、Delete、Update
if(F = =0)
{
T MD5 = MD5calstring(T) ;//计算源表的数据记录的 MD5 值
L = CompareMD5(T MD5,SMD5) ;
//把计算的 MD5 值与影子表 MD5 进行匹配,然后进行相关操作
if(L = =0) //两条记录完全相同
{
null;
}
else
{
Update(T) ;//修改数据添加到增量表中,标志位置"3"
}
Reservation(T_Pointer + + ) ;
//指针重定位,下移指向源表的下一条记录
Reservation(S_Pointer + + ) ;
//指针重定位,下移指向影子表的下一条记录
}
if(F <0)// T_Pointer 所指记录为新增数据
{
Insert (T) ;//插入添加到增量表中,标志位置"1"
Reservation(T_Pointer + + ) ;
}
        else // S_Pointer 所指记录为要删除记录
{
del(S) ;//删除数据添加到增量表中,标志位置"2"
Reservation(T_Pointer - - ) ;
Reservation(S_Pointer + + ) ;
}
}//结束循环 while
if(T. HasRows)//影子表的指针移动到表尾,源表仍有数据
{
Inser (T) ; //源表剩余数据全部为新增加数据
}
if(S. HasRows)//源表的指针移动到表尾,影子表仍然有数据
{
del(S) ;//影子表剩余数据全部为要删除的数据
}
}//算法结束
```

基于 MD5 算法的改进型影子表法,比较方式是移动指针次数与记录条数呈线性增长,与传统的实现方法 NOT EXISTS 和字符串匹配相比,减少了记录的对比次数;又因为在识别增量数据是否为修改数据时,不用遍历所有属性而只是比较 MD5 值,减少了属性对比次数和时间,因而在修改数据多且属性字段也较多的情况下,更能提高对比效率。

2 算法性能分析

下面来计算本方法的时间复杂度。

用以下公式计算增量识别提取花费的时间 T:

$$T = T_1 \times m + T_2 + T_3 + T_4 \times k + T_5 \times m + \text{TMD5}$$

其中:T_1 为 SQL 语句的解析时间,包括解析提取有序结果集的时间、解析 INSERT INTO 语句的时间;T_2 为提取属性时间(主键);T_3 为提取记录集时间;T_4 为移动指针时间;T_5 为以元组为单位,将增量数据插入增量表中的时间,即 INSERT INTO 语句的执行、写入日志等操作的时间;TMD5 为计算属性字段 MD5 的时间和比较 MD5 的时间;m 为增量数据条数;k 为记录总条数。

分析:其中 T_1、T_2、T_3 主要与所用硬件设备和具体数据库性能相关,T_5 的效率取决于采取的插入方案。所以下面主要对 T_4、TMD5 的时间复杂度进行分析。

1)移动指针总时间 T_4

设原始表为 m 行,影子表为 n 行,均为 r 列;当指针移动一

行,需要时间 t_1,因此移动指针总时间为 $T_4 = (m+n) \times t_1$。

2)MD5 运算时间和比较时间 TMD5

MD5 运算时间主要与 MD5 算法性能有关,现在在传统 MD5 算法流程上,有很多改进后的算法,如文献提到的改进后的算法的运行效率都有了很大的提高,通常处理 1 GB 的数据只需 15 ~ 25 s。也就是说,其 MD5 计算开销要远小于遍历所有属性的时间,当比较的属性字段越多的时候,优势越明显。

MD5 属性值比较时间主要与经过 MD5 算法运算得到的字符串长度有关,在这方面都可以通过异或或者折半 MD5 字段,使其长度变得更短。

3)综合分析

设原始表和增量表分别为 m 行和 n 行,且表中包含 r 个字段项,运算和比较 MD5 字段值时间为 tc,则 F 表示输入的时间复杂度为:

a)在最优情况下时间复杂度

$$B(n) = \min(F(m_1, n_1, r)) = m + n$$

其中 $0 < m_1 \leqslant m, 0 < n_1 < n$。

b)在最坏情况下时间复杂度

$$W(n) = \max(F(m_l, n_1, f)) = (m+n) \times tc$$

其中 $0 < m_l \leqslant m, O < n_1 < n$。

c)渐进时间复杂度

假设每一行数据的对比时间为一个单位时间 t_1,则在两表对比的过程中,指针分别从两表的第一行到最后一行各遍历一次,且中间过程无回溯现象,所以消耗的时间为 $(m+n) \times t_1$,也就是说,在这种前提下,算法的复杂度是 $O(n)$。

由以上分析结果可知,适当地调整记录集对比的顺序,可以提高对比速度;而通过计算属性字段 MD5 值限定了对比字段项的个数和长度对比,减少了对比时间。又因为该算法是线性算法,所以不会因为原始数据的增长而出现对比时间的非线性增长,并且在数据变化较多的情况下会取得更高的效率。

3　性能测试与结果分析

为了验证本文提出的基于 MD5 算法的影子表实现方法的优势,对传统影子表中使用的 NOT EXISTS 方法和基于 MD5 的方法(MD5-ST)分别进行了测试和分析。以某公司数据中心在实际业务需求中需要提取增量数据的表和对应字段为例,分别挑选千级、万级、十万级、百万级的数据表进行测试。为了更好地比对结果,暂不考虑数据插入增量表中的时间(与具体的插入方案有关),即数据入库的时间,只比较检测识别变化记录的时间。

1)测试环境　软件为 Windows 2003 Server、.NET Framework 3.5、C#;硬件为 IBM 服务器,CPU 2.0 GHz,内存 2 GB,硬盘 300 GB;操作系统为 Windows 2003 Server;数据库版本为 Oracle 9.1。

考虑到整个测试过程只是涉及到数据读取与 MD5 值的计算,并无过多的与操作系统、软件平台和开发语言相关的操作。因此可以认为上述测试方法的结果具有普遍性,即也适用于其他操作系统平台(如 Linux/UNIX)或应用语言/平台(C#、C、Java)。

2)测试结果

表 1　测试表格

表名称	记录数	列数	增量比例/%	文本大小/MB
cd_cement	7 515	26	13	1.71
cd_wellbore	17 560	8	11	3.3
cd_well_source	85 754	29	6	29.9
dm_event	771 889	14	15	266
cd_comp_t	467 905	67	9	189

在表 1 中,给出了数据以文本形式存储时所占用的存储空间大小,并给出了增量数据所占的比例,这两项指标对增量数据识别算法的效率影响很大。在图 1 中给出了在该环境下,对这些表数据采用传统影子表法和基于 MD5 的改进型影子表法的运行时间对比。

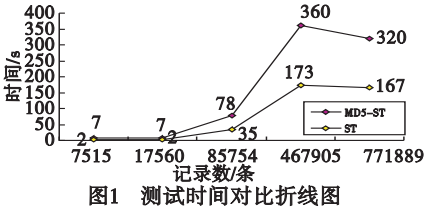

图1　测试时间对比折线图

从折线图可以看出,MD5 影子表法的整体时间趋势比传统的影子表方法时间缩短了,尤其是随着数据量的增加,在数据变化较大的情况下,这种方法优势更明显。

4　结束语

本文针对增量数据提取效率问题,在传统影子表法实现方法的基础上,通过调整记录集的比对顺序,并结合 MD5 算法,提出了融入 MD5 的影子表法。该算法将属性字段变换为 MD5 值,并对源表和影子表进行线性比较,大大降低了数据的比对次数和时间,降低了算法的时间复杂度,提高了数据的比对效率。经应用测试表明,该算法能够快速识别发生变化的记录,提高了增量数据提取效率和访问效率。该算法已在油田数据中心增量数据提取方面得到了应用验证,并取得了良好效果。

参考文献:

[1] 郑祥云,张娟,葛文庚. 数据库同步中差异数据捕获方案设计与实现[J]. 电脑知识与技术,2009,5(7):1544-1545.

[2] 史晶波. 在 DB2 中提取增量数据的一种方法[J]. 计算机与数字工程,2004,32(6):15-16.

[3] 张莹,徐剑,常桂然,等. Ad hoc 网络中双向快速字符串匹配算法[J]. 计算机研究与发展,2010,37(10):42-47.

[4] 丁国强,赵国增,李传锋. 改进 BM 算法策略的网络入侵检测系统设计[J]. 计算机测量与控制, 2011,19(11):2661-2664.

[5] 敖莉,舒继武,李明强. 重复数据删除技术[J]. 软件学报,2010,21(5):917-929.

[6] 李懿,罗军. 影子主码及其在工作流引擎设计中的应用[J]. 计算机工程与应用,2006,42(32):158-159,166.

[7] 周林,韩文报,祝卫华,等. MDx 差分攻击算法改进及 GPGPU 上的有效实现[J]. 计算机学报,2010,33(7):1177-1181.

[8] KI LEE Y, CHAN H, VERBAUWHEDE I. Design methodology for throughput optimum architectures of hash algorithms of the MD4-class [J]. Signal Processing Systems,2008,53(1-2):89-102.

[9] ALGREDO-BADILLO I, FEREGRINO-URIBE C, CUMPLIDO R, *et al*. Design and implementation of a non-pipelined MD5 hardware architecture using a new functional description [J]. IEICE -Trans on Information and Systems,2008,E91-D(10):2519-2523.

[10] 杜昌钰. MD5 算法的过程分析及其 C#实现[J]. 通信技术,2008,41(8):71-72.