列存储数据库关键技术综述

李 超 张明博 邢春晓 胡劲松
(清华大学信息技术研究院 北京 100084)

摘 要 随着互联网技术的发展、硬件的不断更新、企业及政府信息化的不断深入,应用的复杂性要求越来越高,推动着数据存储技术向着海量数据、分析数据、智能数据的方向发展,以便为数据仓库、在线分析提供高效实时的技术支持。基于行存储的数据库技术面临新的问题,已经出现了技术瓶颈。近些年来,一种新的数据存储理念,即基于列存储的关系型数据库(简称列数据库,下同)应运而生。列数据库能够快速发展,主要原因是其复杂查询效率高,读磁盘少,存储空间少,以及由此带来的技术、管理和应用优势。对列数据库技术的基本现状、关键支撑技术以及应用优势进行了介绍和分析。

关键词 列数据库,列存储,数据压缩,延时物化,成组迭代,不可见连接,数据仓库,商业智能,TPCH
中图法分类号 TP391 **文献标识码** A

Survey and Review on Key Technologies of Column Oriented Database Systems

LI Chao ZHANG Ming-bo XING Chun-xiao HU Jin-song
(Research Institute of Information Technology, Tsinghua University, Beijing 100084, China)

Abstract Column-oriented database is a kind of new database storage technology that stores data according to column (not traditionally row). The database pioneers such as Dr. Michael Stonebraker are advocating and exploring the new theory and technology for column-oriented database. The main features of it are good query efficiency, less disk access, less storage, and significant improvement of database performance. Column-oriented database is an ideal architecture for data warehouse natively, and thus shows a good potential in supporting highly efficient business intelligence applications. This new technology is promising in both academic and business, therefore attracting lots of high-tech corporations and research institutes to devote in it. This paper introduced and analysed the main features, key technologies and current R&D situations of column-oriented database.

Keywords Column-oriented database, Compression, Block iteration, Late materialization, Invisible join, Data warehouse, Business intelligence, TPCH

1 引言

列数据库是基于列存储的、主要面向企业决策分析领域的关系型数据库。在SIGMOD85,论文"A Decomposition Storage Model"[1]提出了一种新的存储概念,简称DSM,这就是列数据库的雏形,但是这种技术在当时并没有得到足够的重视。近些年来在以Michael Stonebraker, Daniel J. Abadi, Peter Boncz为首的一批专家的大力提倡下,列数据库相关技术及应用快速发展,在企业决策领域已经开辟了一条新道路(参考网址 www.databasecolumn.com)。这种技术的特点是复杂数据查询效率高,读磁盘少,存储空间少。这些特点使其成为构建数据仓库的理想架构,因而引起数据库学术前沿和相关高新科技企业投入大量的人力和物力研发。

1.1 列数据库基本概念

列数据库是对应并区别于行数据库的概念。行数据库就是我们所熟知的传统关系型数据库,即数据按记录存储,每一条记录的所有属性都存储在一起,如果要查询一条记录的一个属性值,需要先读取整条记录的数据。而列数据库是按数据库记录的列来组织和存储数据的,数据库中每个表由一组页链的集合组成,每条页链对应表中的一个存储列,而该页链中每一页存储的是该列的一个或多个值。

1.2 列数据库的学术价值与应用价值

列数据库技术有它独有的学术价值,近些年来在国际一流的数据库会议上频频有关于这个领域的优秀论文出现[1-3,5-17],他们主要围绕其商业价值以及主要关键技术,包括基于其主要存储原理的存储压缩、延时物化、成组叠代、查询优化、索引、及加密等进行研发。

列数据库的应用价值来自它对复杂查询的灵活快速以及压缩所带来的存储优势,这使其在数据仓库和商务智能方面具有良好的发展前景。已经有许多列数据库在企业决策分析

到稿日期:2010-01-08 返修日期:2010-03-22 本文受国家863计划(编号2009AA01Z143),铁道部-清华大学科技研究基金(编号:J2008X009)资助。

李 超(1978—),女,博士,讲师,主要研究方向为存储技术、数据库技术等,E-mail:li-chao@tsinghua.edu.cn;张明博(1982—),男,工程师,主要研究方向为Web信息管理、数据库技术等;邢春晓(1967—),男,博士,教授,主要研究方向为数据库技术、数字图书馆等。

领域的成功案例。VERTICA 已经在美国拥有了许多客户，SYBASE IQ 更是已经进入了中国市场。

1.3 主要的开源列数据库和商业列数据库

1.3.1 C-Store

C-Store 是一款开源的、运行于 Linux 系统的列数据库，是耶鲁大学、麻省理工大学、布朗大学等联合协作开发的软件，于 2006 年 10 月发布。它是一种适合做学术研究的列数据库，目前主要的学术研究都是利用它来做的，在学术界比较流行。它同时是商业列数据库产品 VERTICA 的原型（http://db.csail.mit.edu/projects/cstore/#people）。

1.3.2 MonetDB

MonetDB 是一款运行于 Linux 和 Windows 系统上的高性能开源列数据库（http://monetdb.cwi.nl/），同时是一款内存数据库。它可应用于数据挖掘、OLAP、GIS、XML 查询、文本和多媒体检索。它支持 SQL'99，SQL'03 核心标准，支持持久存储、触发器。用户可用 C 编写所需功能。它还支持 OPEN-GIS 标准和 SQL/XML 的大部分标准。它基于内存文件存储，可对数据库进行升级，支持 32 位和 64 位平台，能对查询进行有效的优化[14]。

1.3.3 MonetDB/X100

MonetDB/X100 同 MonetDB 都是一个组织开发的，主要区别是 X100 不是内存数据库，而其技术也更加成熟一些，查询效率也更加优秀。MonetDB/X100 同样运行于 Linux 和 Windows 系统上，同样由 C 编写。

1.3.4 Rasdaman

Rasdaman 是一款运行在 Linux 系统上的商业列数据库，由不莱梅大学和 Rasdaman 公司合作开发（www.rasdaman.com），2008 年 9 月宣布开源版本（www.rasdaman.org），全名为"raster data manager"。它是一款快速、灵活、价廉的列数据库，其开发语言是 C/C++以及 JAVA，支持大部分 SQL 标准。

1.3.5 Sybase IQ

SYBASE IQ 是 Sybase 公司专为分析型应用与数据仓库而设计的，是唯一一个由传统基于行存储的关系型数据库厂商开发的列数据库产品。Sybase IQ 是拥有列式存储、专利索引、查询优化等技术的数据仓库引擎，带来的查询速度将比传统数据库提升 10～100 倍。

1.3.6 ParAccel

ParAccel 分析数据库 PADB 是一个专门开发的数据仓库和分析型数据库管理系统，ParAccel 具有可扩展性、简单的数据仓库的安装和操作、成本可接受性、高效解析查询处理等特点。

除此之外，还有 Vertica，X100/VectorWise，KickFire，SAP Business Accelerator，Infobright，Exasol 等开源列数据库以及商业列数据库。

可以看到，列数据库已经从 SIGMOD85 上不为大众所关注的 DSM 雏形发展到了今天初具规模的局面。无论是在学术前沿研究、系统技术研发，还是在数据仓库、数据分析及决策支持等应用领域，列数据库都是一个蓬勃发展的热点和新的增长点。

本文从 3 个方面分析列数据库相对于行数据库的优势；然后介绍列数据库的几大关键技术，包括压缩、延时物化、成组迭代、不可见连接，这既是当前的一些研究前沿和热点，又是不同列数据库系统之间的主要技术区别所在；接下来，介绍用于评价数据库系统性能的 TPCH 测试，以为相关研究提供参考，并引用 TPCH 官方测试结果进一步说明列数据库的优势；最后总结全文。

2 列数据库的优势

列数据库在数据仓库、商务智能领域应用中有着先天的优势：独特的存储方式，能够迅速地执行复杂查询；列数据库的压缩技术，更是能为数据仓库、商务智能应用中巨大的数据量节约存储成本；列数据库先进的索引技术也大大提高了数据库的管理。

2.1 列数据库存储方式带来的技术优势

因为列数据库和行数据库都是关系型数据库，因此列数据库在逻辑上与行数据库没有区别，用户处理和操作的都是一行一行的记录、一个一个的表。两者的本质区别在于其物理存储是基于列存储还是基于行存储。列数据库按列存储的结构，便于在列上对数据进行轻量级的压缩，列上多个相同的值只需要存储一份。压缩能够大量地降低存储成本。

按列存储和压缩的特点，也为列数据库在查询方面带来先天优势，因为若能将更多的数据压缩在一起，则在每次读取时就可以获得更多的数据。很显然，每次读取操作获取更多的数据就意味着更快的处理速度。同时，列数据库的存储特点有利于迅速查询所需要的列。行存储虽然可以比较轻松地添加修改记录，但是会增加许多不必要的读取；而列存储只需要读取相关数据，并且可以从多个入口写入数据。

2.2 列数据库在数据管理方面的便捷优势

在数据管理方面，行数据库采用的是稠密索引，即当数据库文件中的记录不按照关键码的顺序排列时（比如按照加入的顺序排列），需要对每一个记录建立一个索引项。这有两方面的缺点：一是增加所用的存储空间，二是增加数据更新时的代价。正是由于这两方面的问题，在基于行存储的关系型数据库中，为表的所有列都建立索引就不太现实。这样就出现了下面的问题：如果一个查询语句是基于一个未加索引的列查询，系统就不得不做全表扫描，导致数据库的查询效率不高。因此，考虑在哪些列加索引并根据应用的需求适时调整，就成为数据库管理员的一项繁重工作。而列数据库因为各条记录在磁盘中是按照关键码值压缩顺序存放的，所以采用的是稀疏索引，即把连续的若干记录分成组（块），对一组（块）记录建立一个索引项。列数据库可以为所有列建立稀疏索引，事实上也都是这么做的。这是因为每个列的值都已被压缩顺序存储，索引只需建立到数据块级，因此索引的存储空间很小，维护费用很低，使得可以以很小的代价给所有的列建立索引。当查询通过索引定位到某一数据块后，就可以使用二分法查找。这样，数据查询在任何情况下都不会导致全表扫描，从而提高了数据库的查询性能。在列数据库中，后台程序默认地自动为所有的列维护稀疏索引，因此为数据库管理员卸下了建立、管理和维护索引的繁重工作。

此外，无论是在行数据库还是列数据库中，在删除一条记录时，都会出现一个物理存储空间上不连续的空洞。在行数据库中，随着一段时间的增删操作，这些空洞会越来越多，越来越大。这一方面会导致物理存储空间的闲置和浪费，另一

方面会使得访问数据库的效率下降。因此，行数据库管理员为了填补这些空洞、消除空洞带来的负面影响，经常在维护时要做的事情是把数据全部导出来，再重新导回去。而列数据库因为在每一列上都采用了轻量的稀疏索引，在插入删除数据时，利用这些索引可以把空洞尽量减小，免除了数据库管理员的大量导入、导出工作。

2.3 列数据库在数据挖掘领域的应用优势

数据挖掘应用有着数据量大、主要针对少数列进行复杂查询操作的特点。在列存储模式下，对于列的 DML(Data Manipulation Language，数据操纵语言命令)操作，仅仅是对列所对应的数据库页链进行数据扫描，不会导致对全表的数据访问，可以有效降低 DML 操作的 I/O 数量。由于数据按列存储，为调整数据模型所做的新增列、删除列操作不再会遇到数据碎片问题。这是因为在列存储中，由于数据存储以列为单元，列删除或者新增操作是将列所对应的数据库页链从表的页链集合中去除或者新增，不涉及其他列对应的数据库页链。此外，列数据库中数据按列压缩的特点也同样能够减少挖掘时的 I/O 量[2]。

3 列数据库的关键技术

3.1 压缩

随着信息化的不断发展，在数据仓库与商业智能领域中，数据量已经越来越大，企业不可能不间断地增加存储成本，所以数据压缩已经是目前 IT 领域一个一定要面对的问题。列数据库的存储特点，决定了其在压缩方面的优势。

在 SIGMOD06 会议上，Daniel J. Abadi[3] 通过在开源的列数据库 C-Store 上进行实验比较和理论分析指出，主要的压缩方法有以下几类：

1)行程编码算法(Run-length Encoding)；

2)词典编码算法(Dictionary Encoding)；

3)位向量算法(Bit-Vector Encoding)。

3.1.1 行程编码算法(Run-length Encoding)

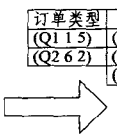
行程编码算法(见图 1)是比较适合列数据库的压缩算法之一，即用一个三元组记录数据值、数据出现的起始位置和持续长度(即行程)，以代替具有相同值的若干连续原始数据，使三元组的存储长度少于原始数据的长度。

三元组描述为(X,Y,Z)：

X:数据的值，Y:起始位置，Z:长度。

以图一为例

订单类型	产品ID	产品类型
Q1	1	2
Q1	1	5
Q1	1	8
Q1	1	4
Q1	2	5
Q2	3	2
Q2	3	4



订单类型	产品ID	产品类型
(Q1 1 5)	(1 1 4)	2
(Q2 6 2)	(2 5 2)	5
	(3 7 1)	8
		4
		5
		2
		4

图 1 行程编码算法

3.1.2 词典编码算法(Dictionary Encoding)

词典编码就是生成一个“原始值-替代值”的对照词典。为了起到压缩的作用，替代值的长度小于原始值的长度。存储的时候，只存储替代值而不是原始值，从而压缩了存储空间。

算法描述如下：

T1:原始表(其中，T1 有 n 行)

DIC:词典表(关于表 T1 的第 k 列的词典)

T2:压缩存储表

N:原始值

M:替代压缩值

For($1 \leqslant i \leqslant n$)

Begin

扫描 T1 中第 k 列的第 i 个值得到原始值 N；

查找 DIC 中 N 对应的压缩值 M；

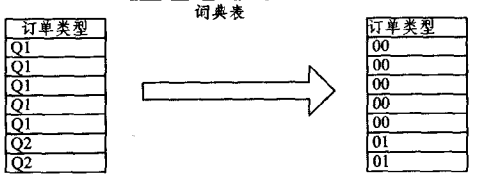
将压缩值 M 作为 T2 中第 k 列的第 i 个值存储；

End.

如图 2 所示的例子中，简单的两位数字代替了原始的字符串，缩短了所需的存储空间长度。

Q1	00
Q2	01
Q3	02
Q4	03

词典表



订单类型
Q1
Q1
Q1
Q1
Q1
Q2
Q2

订单类型
00
00
00
00
00
01
01

图 2 词典编码算法

3.1.3 位向量编码算法(Bit-Vector Encoding)

位向量编码是为每一个不同的取值生成一个位向量，根据位向量(串)中不同的位置取值 0 或 1 来对应并确定不同的原始值。

算法描述如下：

T1:原始表中的某一列(其中，T1 有 n 行)

V:T1 的取值空间(无重复元素)

M:原始值

m:V 的模数(V 中的值的个数)

BVi($1 \leqslant i \leqslant m$):$V$ 中不同取值对应的位向量

For($1 \leqslant i \leqslant n$)

Begin

扫描 T1 中的第 i 个值 M；

将 M 加入取值空间 V；

End;

计算 V 的模 m；

For($1 \leqslant i \leqslant m$)

初始化位向量 BVi；

//m 个位向量分别对应 m 个不同的取值

For($1 \leqslant i \leqslant n$)

Begin

扫描 T1 中的第 i 个值 M；

确定 M 对应的位向量 BVx；

将位向量 BVx 的第 i 位值置为 1；

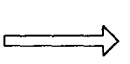
For($1 \leqslant j \leqslant m$)

if($j <> x$) then 将位向量 BVj 的第 i 位值置为 0；

End.

图 3 中“产品 ID”列的原始数据经过位向量算法压缩后，形成如图 2 所示的位向量。

订单类型
Q1
Q1
Q1
Q1
Q1
Q2
Q2



Q1位向量	Q2位向量
1	0
1	0
1	0
1	0
1	0
0	1
0	1

图 3 位向量算法

3.1.4 3种压缩方法比较

Daniel J. Abadi 在文献[5]中介绍的3种压缩方法都是轻量级压缩算法。轻量级压缩最大的优点是能够在不解压的情况下直接对压缩状态的数据进行操作[19]。就目前的列数据库而言，行程编码是一种最容易实现的压缩方法，但是有一定的局限性。表1是3种压缩方法的优缺点比较。

表1 压缩方法比较

算法名称	行程编码算法	词典编码算法	位向量算法
数据列的共同特征	a)适用于重复数据较多 b)不适用于重复数据较少		
适用数据列的不同特征	重复数据的排序比较规则	取值空间较小	取值空间较小
不适用的数据列不同特征	排序不规则	a)取间空间较大 b)数据类型长度比词典符号长度更小	取值空间较大
优点	对于适用的数据特征，有比较好的压缩效果	a)对于数据类型要求较低 b)对于数据排序要求较低	a)对于数据类型要求较低 b)对于数据排序要求较低 c)在有些情况，查询效率要比词典编码高
缺点	对列值的重复性以及排序要求较高	需要创建一张词典表，增加维护代价，如果数据重复性不高，词典表会过于巨大	用位置代表数据，如取值空间较大，或重复性较低，占用空间会比较大

注：表中所谓“适用”即采用了某一压缩算法可以起到减少存储空间的效果。

3.1.5 压缩优势的实例

对于列数据库主要应用的数据仓库领域，海量数据的有效压缩是一个十分重要的优势。几种开源和商业的列数据库系统都将良好的压缩率作为自己突出的优势。列数据库的领军厂家 SYBASE IQ 就声称其压缩率可达到 70%。

表2是2005年 Winter Corporation 做的10大数据库评比。表中的数据仓库大小是建立两年后数据仓库的大小，而原始数据大小是指数据仓库建立时的大小。Nielsen Media Research 是美国一家从事媒体收视率分析研究的公司，基于 Sybase IQ 系统建立了数据仓库，原始大小是 17.969TB。数据仓库建立两年后，大小是 17.685TB。Yahoo 点击流分析数据仓库(基于 Oracle 系统)建立的时候数据量是 17.014 TB，而建立两年后，数据量大小为 100.386TB。基于行数据库 Oracle 的 Yahoo 点击流分析数据仓库从约 17TB 扩张到了约 100TB，大约扩张了 6 倍。而基于列数据库 Sybase IQ 的 Nielsen Media Research 数据仓库却随着数据的积累被压缩得比原来还小。所以，从这个例子可以看出列数据库 Sybase IQ 的压缩比是非常高的。

表2 对比表

公司、组织	数据仓库大小(TB)	原始数据大小(TB)	数据行数	数据库	操作系统	体系结构	数据库厂商	系统厂商	存储厂商
Yahoo	100.386	17.014	3853	Oracle	unix	Centralized /SMP	Oracle	Fujitsu Siemens	EMC
Nielsen Media Research	17.685	17.969	5024	Sybase IQ	unix	Centralized /SMP	Sybase	Fujitsu Siemens	EMC

3.1.6 总结

在国内，已经有同行对压缩算法的性能分析做了很多工作[18]，但列数据库领域的压缩算法研究仍是我们未来工作的重点之一。压缩技术不仅可以节约存储空间，而且是提高查询效率的关键因素。但是上述列数据库压缩方法都有其局限性，所以研发一个适用范围更加广泛、压缩以及解压的时间更加快速的压缩方法，必然是列数据库领域学术研究的一个热点。

3.2 延时物化

为了说明延时物化，先介绍元组物化的概念。元组物化，即将常用元组或可能会用到的逻辑元组从实际物理存储的状态生成为实体化的元组，也称为物化，存储在内存中。在随后查询时，直接读取已经物化的元组，以提高查询的效率。而元组物化有两种方案，分别是提前物化：在提交查询之前物化元组；延时物化：尽量推迟物化元组的时间，在查询中间的某个时刻物化元组。

对于列数据库来说，提前物化需要解压所有已经压缩的数据，其时间和空间的开销是很大的。同时，提前物化会涉及到很多不必要的列，有悖列数据库按列存储、按需取用的初衷。因此，在列数据库领域，提出了延时物化的思想。

3.2.1 延时物化形式化描述

假设有如下查询：

select $value_1, value_2, \cdots, value_N, AGG_1, AGG_2, \cdots, AGG_Q$

from $table_1, table_2, \cdots, table_M$

where $condition_1\ OP_1\ conditon_2\ OP_2, \cdots, OP_T\ condition_P$

$LIST_1, LIST_2, \cdots, LIST_L$

$Value_i(1 \leqslant i \leqslant N)$：查询直接要得到的值

$AGG_i(1 \leqslant i \leqslant Q)$：对于查询的值进行聚集计算的结果

$table_i(1 \leqslant i \leqslant M)$：查询所涉及到的表

$condition_i(1 \leqslant i \leqslant P)$：查询的选择条件

$OP_i(1 \leqslant i \leqslant T)$：逻辑运算(与或非)

$LIST_i(1 \leqslant i \leqslant L)$：对结果进行排序处理指令

算法描述如下：

For($1 \leqslant i \leqslant P$)
  Begin
    对于 $condition_i$，生成标定符合条件的位向量 h_i；
  End;
初始化目标位向量 H；
$H_1 = h_1$；
For($1 < i \leqslant P$)
  $H = H\ OP_{i-1}\ h_i$；
初始化目标物化结果空间为 O；//O 有 $(N+Q)$ 个存储空间
For($1 \leqslant i \leqslant (N+Q)$)
  Begin
    if($i \leqslant N$) then O_i=($Value_i$ 对应的列) AND H
    else O_i=(AGG_{i-N}对应的列) AND H;
  End;
初始化最终查询结果空间为 S；//S 有 $(N+Q)$ 个存储空间

For($1 = < i < = (N+Q)$)
  Begin
    if($i < = N$) then $S_i = O_i$
    else $S_i = AGG_{i-N}(O_i)$;
  End;

输出结果时按照 $LIST_1, LIST_2, \cdots, LIST_L$ 的要求依次排序。

3.2.2 延时物化测试实例

以下面的查询为例：

Select custID,SUM(price) From table where

prodID =4 and storeID=1

group by prodID

如图 4 所示，如果采用提前物化，列数据库在执行查询时，会首先将列中数据解压，之后构建(物化)元组。这样的操作有 3 个缺点：一是需要解压所有数据；二是需要物化所有元组；三是增加内存负担。

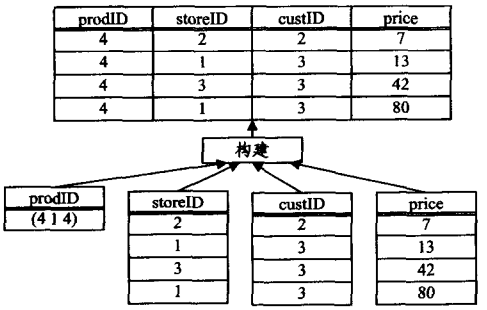


图 4　提前物化

如果采用延时物化，列数据库则按过程执行查询。如图 5 所示，根据判定条件 prodID = 4，storeID = 1，将 prodID，storeID 两列分别用位向量进行选择标定。

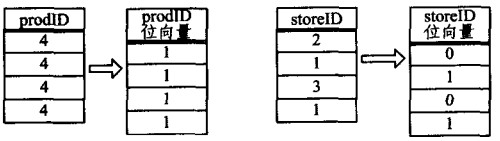


图 5　延时物化第一步

之后将两个列对应的向量进行逻辑与运算，得到一个列向量，如图 6 所示。

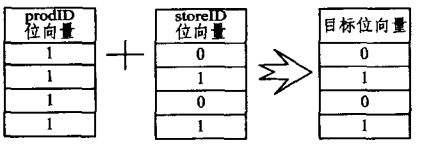


图 6　延时物化第二步

通过上面得到的位向量，与相关两列进行逻辑与运算。在此刻将查询得到的元组进行物化，如图 7 所示。

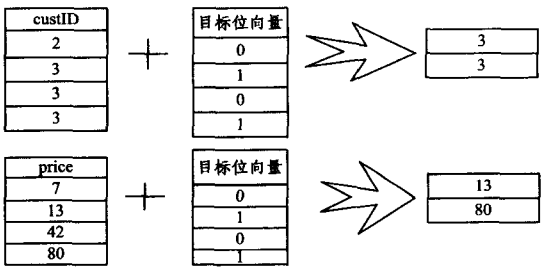


图 7　延时物化第三步

最后通过物化的元组，得到所要查询的结果 custID，SUM(price)，如图 8 所示。

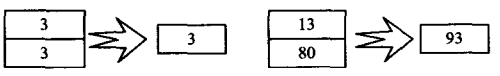


图 8　延时物化第四步

ICDE2007 会议上，Daniel J. Abadi[7] 在对延时物化和提前物化进行测试后得出了令人信服的定量化结论：在低选择性、中选择性和高选择性 3 类具有代表性的查询中，延时物化的表现始终要好于提前物化，如图 9 所示。

在对比测试中，即使在没有压缩的情况下，延时物化的表现仍然要比提前物化的好，如图 10 所示。

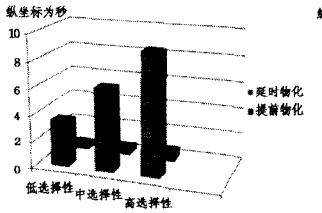


图 9　对比图

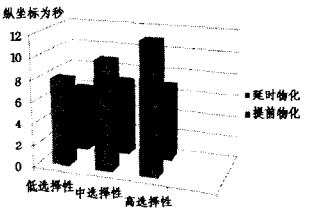


图 10　对比图

3.2.3　总结

延时物化最主要的优点在于其高效的压缩传输数据开销：在执行计划中用位图(位向量)来标识行的位置，直到最后必须取属性值时再实际取相应列的值，尽可能地避免了不必要的实际数据传输开销。延时物化是一个比较实用的技术，非常适合在列数据库中使用。单纯的列数据库是一列一列地把数据存储在磁盘上，如果离开列数据库，延时物化许多性能上的潜力得不到实现[7]。Daniel J. Abadi 在文献[7]中，利用开源列数据库平台 C-Store，详细介绍了两种物化方式的实验过程。另外，MonetDB/X100 这个比较有影响力的开源列数据库也运用了延时物化技术。

3.3　成组迭代

要处理一系列记录，行数据库要对每个记录依次进行迭代，对于每一个记录通过单个记录的操作接口，从这些记录中选取需要的属性或者执行函数的调用。这是一个成本很高的操作。所以 IBM 的专家 S. Padmanabhan 提出，可以成组地调用函数，这样就可以节约很多资源。

对于现代 CPU 而言，CPU 在缓存中找到有用的数据被称为命中；当缓存中没有 CPU 所需的数据时(未命中)，CPU 才访问内存。所以，CPU 访问内存的频率越低(也称未命中频率)，查询的效率也就越高。列式存储具有高度的可压缩性。假设对一列使用行程编码压缩，就像本文 3.1 节中提到的一样(值、起始位置、长度)，占 10 个字节。利用 64 字节的高速缓存行，可以将 6 个压缩的列值载入一个高速缓存行。这样，每次访问内存，读取的压缩列值是 6 个，而这些压缩的列值对应的实际数据就远远不止 6 个。

此外，如果列被设定为固定宽度，这些值可以直接对应为一个数组。把数据当作一个数组来操作，可以使单个记录处理代价最小化。所以，列数据库的存储方式可以大大提高 CPU 吞吐量。随着 CPU 性能的不断提高，运用成组迭代的列数据库在未来的表现将更加优秀[6]。

成组迭代的思想是针对行数据库提出的，所以成组迭代同样也适用行数据库。在这里也可以感觉到，列数据库作为关系型数据库，在许多方面是不得不借鉴行数据库已经成熟的技术。

3.4　不可见连接

这是 Daniel J. Abadi[3] 提出的一种专门针对列数据库进行查询效率优化的技术。该方法将查询所涉及的各个表的属性对于查询条件的符合情况采用位向量的方式来标定，之后对这些位向量进行逻辑与或运算，得到最终可用于标定结果的位向量。整个过程没有属性或列之间直接的值连接操作，这些直接的操作被位向量直接的逻辑与或运算替代，因此被称为“不可见连接”。不可见连接的目的是尽量避免原始数据

传输、处理和缓存的开销，而充分利用按列存储的便利，采用位向量的方式来标定和连接符合条件的中间结果。该方法适用于星状结构的数据库。

3.4.1 不可见连接形式化描述

假设有如下查询：

select $value_1, value_2, \cdots, value_N, AGG_1, AGG_2, \cdots, AGG_Q$

from $table_1, table_2, \cdots, table_M$

where

$condition_1\ OP_1\ conditon_2 OP_2 \cdots OP_T condition_P$

$LIST_1, LIST_2, \cdots, LIST_L$

$Value_i(1 \leqslant i \leqslant N)$：查询直接要得到的值

$AGG_i(1 \leqslant i \leqslant Q)$：对于查询的值进行聚集计算的结果

$table_i(1 \leqslant i \leqslant M)$：查询所涉及到的表

$condition_i(1 \leqslant i \leqslant P)$：查询的选择条件

$OP_i(1 \leqslant i \leqslant T)$：逻辑运算(与或非)

$LIST_i(1 \leqslant i \leqslant L)$：对结果进行排序处理指令

算法描述如下：

count=0；//count 用于记录选择性查询条件的个数

For($1 \leqslant i \leqslant P$)

Begin

if($condition_i$ 是选择性条件而非连接条件)

then

Begin

对于 $condition_i$，生成标定符合条件的位向量 h_i；

count=count+1；

end；

End；

初始化目标位向量 H；

$H_1 = h_1$；

For($1 < i \leqslant P$)

if($condition_i$ 是选择性条件而非连接条件)

then $H = H\ OP_{i-1}\ h_i$；

初始化中间物化结果空间 O；

//O 有(P-count)个存储空间将被物化中间结果填充

For($1 \leqslant i \leqslant P$)

Begin

if($condition_i$ 是连接条件而非选择性条件)

then O_i=($condition_i$ 对应的事实表的列) AND H

End；

初始化最终连接结果空间为 R

//R 有(P-count)个存储空间将被最终连接结果空间填充

For($1 \leqslant i \leqslant P$)

Begin

if(O_i 存储的是中间物化结果而非初始化值)

then

Begin

$condition_i$ 对应的维表与 O_i 在对应列作连接；

R_i=选取连接结果中与 $Value_i(1 \leqslant i \leqslant N)$ 或者 $AGG_i(1 \leqslant i \leqslant Q)$ 关联的列；

End；

End；

初始化最终查询结果空间为 S //S 有($N+Q$)个存储空间

For($1 \leqslant i \leqslant (N+Q)$)

Begin

if($i \leqslant N$) then $S_i = R_i$

else $S_i = AGG_{i-N}(R_i)$；

End；

输出结果时按照 $LIST_1, LIST_2, \cdots, LIST_L$ 的要求依次排序。

3.4.2 不可见连接测试实例

以下面的查询为例：

```
select c_nation, s_nation, d_year, sum(lo. revenue) as revenue
from customer AS c, lineorder AS lo, supplier AS s, dwdate AS d
where lo_custkey=c_custkey
and lo_suppkey=s_suppkey
and lo_orderdate=d_datekey
and c_region='ASIA'
and s_region='ASIA'
and d_year ≥ 1992 and d_year ≤ 1997
group by c_nation, s_nation, d_year
order by d_year asc, revenue desc;
```

首先我们按照这个顺序来执行 where 子句，lineorder，customer(region=Asia)，supplier(region=Asia)，date(year between 1992 and 1997)。

从表 3 中，根据判定条件 region=Asia，得到 custkey 为 1 和 3 的记录符合条件。

表 3 客户表

custkey	region	nation
1	ASIA	CHINA
2	EUROPE	FRANCE
3	ASIA	INDIA

在表 4 中，根据判定条件 region=Asia，得到 suppkey 为 1 的记录符合条件。

表 4 供应商表

suppkey	region	nation
1	ASIA	RUSSIA
2	EUROPE	SPAIN

在表 5 中，根据判定条件(d_year≥1992 and d_year≤1997)，3 条记录都符合条件。

表 5 日期表

dateid	year
01011997	1997
01012997	1997
01013997	1997

之后在 lineorder 表(fact table)(见表 6)中根据上述判定条件进行符合条件的连接。

表 6 事实表

orderkey	custkey	suppkey	orderdate	revenue
1	3	1	01011997	43256
2	3	2	01011997	33333
3	2	1	01021997	12121
4	1	1	01021997	23233
5	2	2	01021997	45456
6	1	2	01031997	43251
7	3	2	01031997	34235

如图 11 所示，根据对表 3－表 5 的选择判定，将表 6 中 custkey，suppkey，orderdate 这 3 个用于连接的列分别用位向量进行选择标定，之后将这 3 个列对应的向量进行逻辑与运算，得到一个用于传递实际选择条件和连接结果的列向量，如图 12 所示。

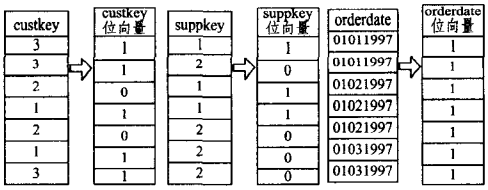


图 11　不可见连接第一步

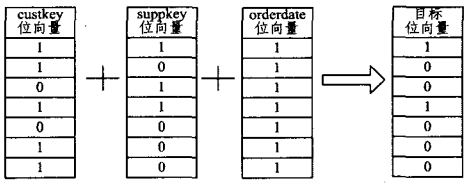


图 12　不可见连接第二步

通过不可见连接，就可以得到所要查询的数据，revenue (43256+23233)，c_nation 为(china，india)，s_nation 为(Russia，Russia)，d_year 为(1997，1997)，如图 13 所示。

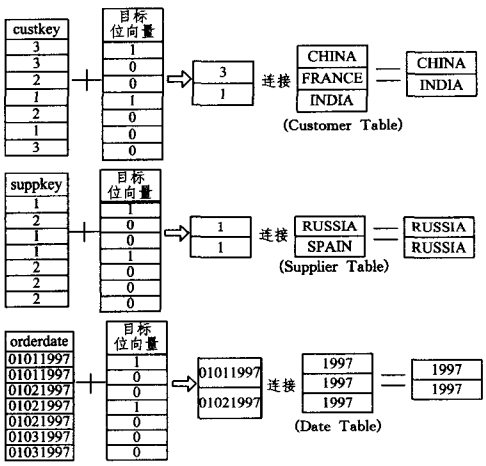


图 13　不可见连接第三步

3.4.3　小结

不可见连接是列数据库专家 Daniel J. Abadi 在 2008 年提出的技术。该技术还非常年轻，还没有运用到真正的列数据库系统上，所以需要通过实践来验证它存在的价值。数据库连接无疑是数据库技术中非常重要的组成部分，所以针对列数据库连接方面的研究将成为列数据库领域工作者未来工作的重点之一。

3.5　总结

综上所述，支持列数据库的 4 大关键技术分别是压缩、延时物化、成组迭代和不可见连接。现如今，已经有很多列数据库成功地实施并运用了这些技术，将来围绕着这 4 个方面，会有许多新技术不断出现，以促进列数据库技术的发展。

4　列数据库性能测试

列数据库的发展势头非常迅猛，大有在数据仓库领域取代传统行数据库的趋势。其两大主要技术优势——有效的压缩和高效的查询性能，使其成为支撑大规模数据仓库和提供高性能数据分析的良好平台。作为衡量两大技术优势之一——高效查询的性能测试标准，TPCH 官方测试目前被广泛认可，其测试结果无论是在产品选型还是实验研究方面都极具参考价值。

4.1　TPCH 官方数据库测试

4.1.1　TPC 测试简介

TPC(Transaction Processing Performance Council，事务处理性能委员会)是由数 10 家会员公司创建的非盈利组织，总部设在美国。该组织对全世界开放，但迄今为止绝大多数会员都是美、日、西欧的大公司。TPC 的成员主要是计算机软硬件厂家，而非计算机用户。TPC 的功能是制定商务应用基准程序(Benchmark)的标准规范、性能和价格度量，并管理测试结果的发布。

4.1.2　TPCH 测试简介

TPC-H(商业智能计算测试)是 TPC 的重要测试标准之一，主要用来模拟真实商业的应用环境。与科学计算测试不同，商务智能计算测试是对现实中商用计算需求的全面模拟，包括模拟真实商业交易数据库的动态查询，作为决策支持与数据库应用系统的参考。

这种商业测试可以全方位评测系统的整体商业计算综合能力，对厂商的要求更高，同时具有普遍的商业实用意义，目前在银行信贷分析和信用卡分析、电信运营分析、税收分析、烟草行业决策分析中都有广泛的应用。

4.1.3　TPCH 测试分析

图 14 是根据 TPC 官方网站上的 TPC-H 测试数据分析完成的柱状图[4]。QphH 为纵坐标，它的含义是每小时完成的查询数，这个值越高，说明查询执行的性能越高。图 14 亦是 TPCH 官方测试 100G 数据级的前 10 名对比柱状图。

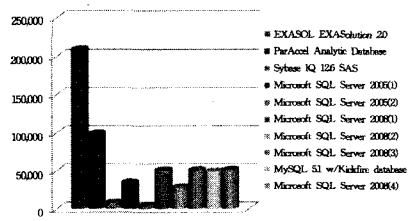


图 14　TPCH 对比图

从图 14 中可以看出，在这一数据级中，表现最好的两个数据库是 EXASQL EXASolution2.0 和 ParAccelAnalytic Database，它们都是列数据库。上述结果说明，列数据库在 100G 这个数据级里表现出色(注：图 14 中，相同的数据库若采用的操作系统不同，其测试结果也就不一样，详细的操作系统及硬件配置参考文献[4])。

4.2　总结

在 TPCH 官方测试中，列数据库已经在查询性能上展示了突出的表现，这进一步印证了列数据库是支撑数据仓库、数据分析以及商务智能的优良平台。列数据库的基于列存储的方式和在此基础上的几大关键技术决定了其适用领域和应用的优势。

结束语　在经过一系列调研与学习之后，我们实验室小组开发了一个列数据库 HUABASE，它虽然只是个内核，但是已经在内部测试中表现出了良好的性能，其效率已经大大超过了进行对比的几个行数据库。

本文介绍了列数据库概念、几个商业列数据库和开源列数据库以及列数据库的优势，着重介绍了列数据库的几大关键技术特点；引用了 TPCH 官方测试结果，最后进行了总结。

(下转第 17 页)

Dynamic Grid Environment[C]// Proceedings of DEXA Workshop. 2005:356-360

[29] Cai M, Frank M, Chen J, et al. MAAN: A Multi-Attribute Addressable Network for Grid Information Services[C]// Proceedings of 4th International Workshop on Grid Computing. Phoenix, USA, 2003:184-191

[30] Andrzejak A, Xu Z. Scalable Efficient Range Queries for Grid Information Services[C]// Proceedings of the 2nd International Conference on Peer-to-Peer Computing. Linkoping, Sweden, 2002:33-40

[31] Oppenheimer D, Albrecht J, Patterson D. et al. Distributed Resource Discovery on Planetlab with SWORD[C]// Proceedings of WORLDS 2004. San Francisco, USA, 2004:9-15

[32] Spence D, Harris T. XenoSearch: Distributed Resource Discovery in the XenoServer Open Platform[C]// Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing. Seattle, USA, 2003:216-225

[33] Balazinska M, Balakrishnan H, Karger D. INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery [C]// Proceedings of the Pervasive 2002. Zurich. Switzerland, 2002:149-153

[34] Bharambe A R, Agrawal M, Seshan S. Mercury: Supporting Scalable Multi-Attribute Range Queries[C]// Proceedings of ACM SIGCOMM 2004. Portland, USA, 2004:353-366

[35] Schmidt C, Parashar M. Flexible Information Discovery in Decentralized Distributed Systems[C]// Proceedings of 12th International Symposium on High-performance Distributed Computing. Seattle, USA, 2003:226-235

[36] Ratnasamy S, Hellerstein J M, Shenker S. Range Queries over DHTs[R]. IRB-TR-03-009. Intel Corporation, 2003

[37] Basu S, Banerjee S, Sharma P, et al. Nodewiz: Peer-to-peer Resource Discovery for Grids[R]. HPL-2005-36. HP Labs, 2005

[38] 张忠平，雷炳银，刘欣媛. 基于多层覆盖网络结构的资源发现机制[J]. 计算机科学，2008，35(3)：103-105

[39] 朱凌，黄德才，郑月锋. 一种基于索引 P2P 分层的网格资源发现模型[J]. 计算机工程与应用，2010，46(2)：96-100

(上接第 7 页)

列数据库在整个数据管理技术领域中的地位如图 15 所示。

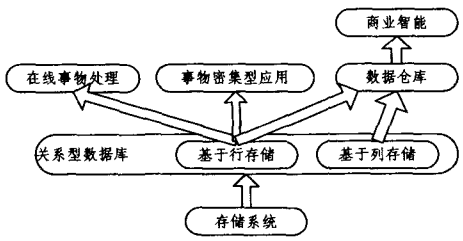


图 15 列数据库在整个数据管理技术领域中的位置

如果 10 年前列数据库只是学术中的一个理论研究的话，如今它已经是一个实实在在的产品，已经在企业决策支持领域使用。不能否认的是，它已经在很多地方展示了自己的特点和优势，已经在竞争激烈的商务智能市场占有一席之地。

列数据库技术还远没有行数据库那么成熟，至今有关它的研究也是数据库领域在学术上一个引人注目的分支，国际上有关的论文频频出现在 SIGMOD, ICDE, VLDB 等重要学术会议上。国内对于列数据库的研究非常欠缺，我们的研究是希望把列存储理念及相关技术引入国内，为我国的数据库领域以及企业决策领域尽一份力量。

列数据库在数据仓库方面有着先天的优势。它即使不会带来数据库领域的革命，也会为商业智能开辟一条新的道路。

参 考 文 献

[1] Copeland G P. A decomposition storage model[C]// SIGMOD '85: Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data. 1985:268-279

[2] 田立中. 列存储在数据挖掘中的应用[J]. 金融电子化，2008(9)

[3] Abadi D J. ColumnStores vs. RowStores: How Different Are They Really? [C]// Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. 2008:967-980

[4] Top Ten TPC-Hby Price / Performance [EB/OL]. http://tpc.org/tpch/results/tpch_price_perf_results.asp

[5] Abadi D J. Integrating Compression and Execution in Column Oriented Database Systems[C]// SIGMOD. Chicago, IL, USA, 2006:671-682

[6] Padmanabhan S, Malkemus T, Agarwal R, et al. Block oriented processing of relational database operations in modern computer architectures[C]// Proceedings of the 17th International Conference on Data Engineering. 2001:567-574

[7] Abadi D J. Materialization Strategies in a Column-oriented DBMS[C]// ICDE. Istanbul, Turkey, 2007:466-475

[8] Abadi D J. Query Execution in Column-oriented Database Systems[C]// SIGMOD. SIGMOD Jim Gray Doctoral Dissertation Award, 2008:145-148

[9] Abadi D J. Column stores for wide and sparse data[C]// CIDR. Asilomar, CA, USA, 292-297

[10] Stonebraker M. C-Store: A Column-oriented DBMS[C]// VLDB. Trondheim Norway, 2005:553-564

[11] Ge Tingjian. Fast, Secure Encryption for Indexing in a Column-oriented DBMS[C]// ICDE. 2007:676-685

[12] Ivanova M. Self-organizing Strategies for a Column-store Database[C]// Proceedings of the 11th International Conference on Extending Database Technology. 2008:157-168

[13] Pranav V. Characterization of TPC-H Queries for a Column-oriented Database on a Dual-core AMD Athlon Processor[C]// Proceeding of the 17th ACM conference on Information and Knowledge Management. 2008:1411-1412

[14] Boncz P A, Kersten M L, Manegold S. Breaking the Memory Wall in MonetDB[J]. Communications of the ACM, 2008, 51(12):77-85

[15] Cornacchia R, Heman S, Zukowski M, et al. Flexible and efficient IR using array databases[J]. VLDB Journal, special issue on IR\&DB integration, 2008, 17(1):151-168

[16] Boncz P A, Grust T, van Keulen M, et al. MonetDB/XQuery: A Fast XQuery Processor Powered by a Relational Engine[C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. Chicago, IL, USA, June 2006

[17] Zukowski M, Boncz P A, Nes N, et al. MonetDB/X100-A DBMS in the CPU Cache[J]. IEEE Data Engineering Bulletin, 2005, 28(2):17-22

[18] 黄鹏，李占山，张永刚，等. 基于列存储数据库的压缩态数据访问算法[J]. 吉林大学学报：理学版，2009(5)

[19] O'Connell S J, Winterbottom N. Performing Joins Without Decompression in a Compressed Database System[J]. ACM SIGMOD Record, 2003, 32(1):6-11