**列式数据库**

前段时间，听了DB2 BLU的讲座，了解了BLU的历史，特性，以及作为列式存储库的各种优缺点，在此借此机会与大家分享一些列式数据库的相关知识，包括列式数据库的历史、存储、优缺点。

**1.概念**

从概念上来说，列式数据库，顾名思义，列式数据库是以列相关存储架构进行数据存储的数据库，主要适合于批量数据处理和即时查询。相对应的是行式数据库，数据以行相关的存储体系架构进行空间分配，主要适合小批量的数据处理，常用于联机事务型数据处理。

举一个简单的例子：

|  |  |  |  |
| --- | --- | --- | --- |
| EmpId | LastName | FirstName | Salary |
| 1 | smith | joe | 40000 |
| 2 | jones | mary | 50000 |
| 3 | Johnson | cathy | 44000 |

这个简单的表包括员工代码（EmpId），姓名字段（Lastname firstname）及工资（Salary）。这个表存储在电脑的内存（RAM）和存储（硬盘）中。虽然内存和硬盘在机制上不同，电脑的操作系统是以同样的方式存储的。数据库必须把这个二维表存储在一系列一维的“字节”中，操作系统再写到内存或硬盘中。行式数据库把一行中的数据值串在一起存储起来，然后再存储下一行的数据，以此类推。

1， Smith，Joe，40000；

2，Jones，Mary，50000；

3，Johnson，Cathy，44000；

列式数据库把一列中的数据值串在一起存储起来，然后再存储下一列的数据，以此类推。1，2，3；Smith，Jones，Johnson；Joe，Mary，Cathy；40000，50000，44000；这是对列式数据库的一个简化的说法。

**2.** **列式数据库与行式数据库**

归根溯源，列式数据库可以追溯到1994年，Sybase公司收购了一家名为ExpressWay Technologies的公司，其主要产品是帮助传统数据库做报表加速的工具，原理就是把行式数据库的数据以列式的方式存储下来。这个技术在1996年被正式推出为产品Sybase IQ，并延续到了现在。我们来看看为什么这样一个从行式到列式的转变可以提高报表的速度。

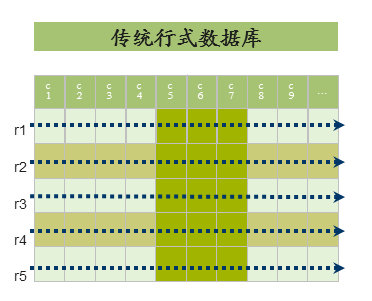


图1，传统行式数据库

行式数据库顾名思义，存储格式是按照“行”的方式把一行各个字段的数据存在一起，一行一行连续存储的，如图1。这样的话，对把一条数据的信息写到数据库中；或者对一条数据中的某些字段进行修改；或者删除整条数据一类的OLTP操作来说既直观也高效。但是，在行式数据库上做一些报表、分析的时候，大家又发现这种存储格式使用效率不高，因为大部分统计分析场景，例如：统计各省份的销售额和利润同比变化；按照部门统计业绩完成情况等等，都是在其中某些字段上的操作，行式数据库不分情况一律按照页面读取数据的方式，在只分析销售额和利润的时候，把每一份合同的其他信息，如客户名称，签约时间，客户经理等等也统统都读了进来，浪费了大量宝贵的I/O。

数据库界给出的第一个改进办法就是“索引”，就像字典前面的目录一样，做到快速定位。但是随着分析场景变得越来越复杂、变化越来越多，DBA们发现索引通常只能为一部分查询、分析起到帮助的作用，如果想为一个企业级的BI系统中所有的查询、分析场景做优化，无论是从组合的角度，还是从开销的角度，都几乎是不可能的，因为大量的索引所带来的存储空间的浪费以及为维护这些索引所带来的时间的浪费都会以指数级别增长。

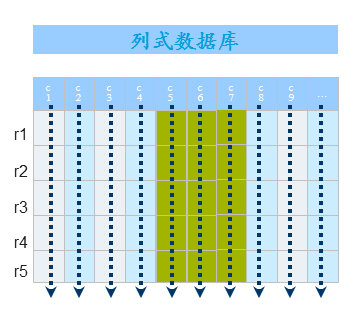


图2，列式数据库

而列式数据库的思路原理并不复杂，把行式数据全部拆开，按照列的方式重新组合存储，一列的所有行的数据存放在一起；按照列内数据的特征值（通常像时间、部门代码、销售地区等维度字段的特征值并不多，几个到几百个很常见）进行高效编码，并且在实际存储中以编码形式存储，这样就带来了大比例的压缩。

带来的好处是：原来只分析销售额的查询就只访问销售额字段，即使是所有历史时期的数据，也不存在读多余的无关数据的问题。

真正的列式数据库具有的创新性在于所有字段都是索引的，甚至可以认为索引和数据是统一的，这样一来，企业数据分析中最困难的随机查询，反而变成了列式数据库的长项，经常出现在这些复杂、多表关联、历史数据分析的场景中比传统的行式数据库快成百上千倍的情况。

列式数据库在数据仓库、数据集市、企业商务智能（BI）等领域已经发挥了越来越多的作用，在全球数以千计的企业中支撑着大量的大数据分析场景的应用，最大的可公开的数据仓库压缩后的数据量达几百TB，约合传统行式数据库内几PB的总量。各家数据库厂商都已经以不同的形式在接受列式存储技术，推出不同风格、不同类型的列式存储的产品，例如最新的微软SQL Server 2012中的Column store Index就是一个很好的例子，可选择地定义个别列式存储的索引。

**3.列式数据库优缺点**

主要优点有下列几点：

1.高效的压缩率，不仅节省储存空间也节省计算内存和CPU；

传统的行式数据库由于每个列的长度不一，为了预防更新的时候不至于出现一行数据跳到另一个block上去，所以往往会预留一些空间。而面向列的数据库由于一开始就完全为分析而存在，不需要考虑少量的更新问题，所以数据完全是密集储存的。行式数据库为了表明行的id 往往会有一个伪列rowid 的存在。列式数据库一般不会保存rowid。

列式数据库由于其针对不同列的数据特征而发明的不同算法使其往往有比行式数据库高的多的压缩率，普通的行式数据库一般压缩率在3：1 到5：1 左右，而列式数据库的压缩率一般在8：1到30：1 左右。

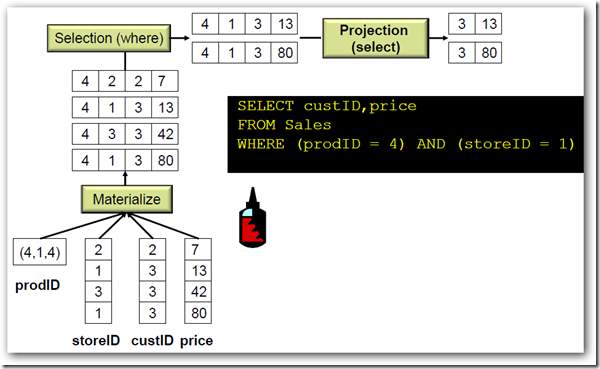
2.不可见索引；

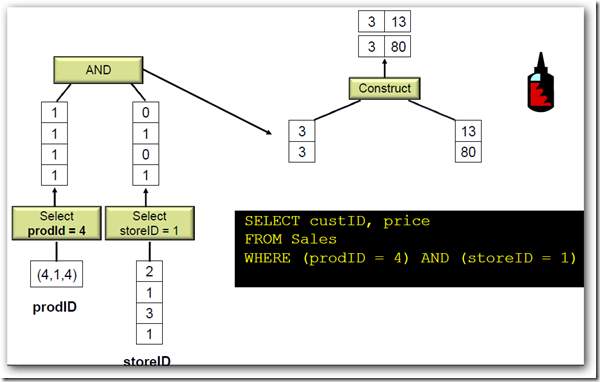
列式数据库由于其数据的每一列都按照选择性进行排序，所以并不需要行式数据库里面的索引来减少IO和更快的查找值的分布情况。如下图所示: 当数据库执行引擎进行where条件过滤的时候。只要它发现任何一列的数据不满足特定条件，整个block 的数据就都被丢弃。最后初步的过滤只会扫描可能满足条件的数据块。

另外在已经读取了可能的数据块之后，对于类似age<65或job=‘Axx’的，列式数据库并不需要扫描完整个block，因为数据已经排序了。如果读到第一个age=66 或者 Job = ‘Bxx’ 的时候就会停止扫描了。这相当与行式数据库索引里的范围扫描。

3.延迟物化；

列式数据库由于其特殊的执行引擎，在数据中间过程运算的时候一般不需要解压数据而是以指针代替运算，直到最后需要输出完整的数据时。如下图：

而行式数据库在运算的一开始就解压缩所有数据，然后执行后面的过滤，投影，连接，聚合操作而列式数据库的执行计划却是这样的，如下图：



4. 极高的装载速度（最高可以等于所有硬盘IO的总和，基本是极限了）

5．适合大量数据，并且非常适合做聚合操作。

目前存在的主要缺点：

1.不适合扫描小量数据。

2.不适合随机的更新。

3.批量更新情况各异，有的优化的比较好的列式数据库（比如Vertica)表现比较好，有些没有针对更新的数据库表现比较差。

4.不适合做含有删除和更新的实时操作。

总有人认为如果每次扫描较多行或者全列全表扫描的时候，行式数据库比列式数据库更有优势。事实上这只是行式数据库认识上的一个误区，即认为列式数据库的主要优势在于其列分开储存，而忽略了列式数据库上面提到的其他几大特征，这个才是列式数据库高性能的核心。