# 面向行 vs 面向列的数据库

Last modified: October 24, 2019 有两种组织关系数据库的方法:

- 面向行
- **面向列**(也称为**列式**或 C-store)

**面向行的数据库**是按记录组织数据的数据库,将与记录关联的所有数据在内存中彼此相邻。面向行的数据库是组织数据的传统方式,仍然为快速存储数据提供了一些关键优势。

**面向列的数据库**是按字段组织数据的数据库,所有与字段关联的数据都在内存中彼此相邻。列 式数据库已变得越来越流行,并为查询数据提供了性能优势。

### 面向行的数据库

传统的数据库管理系统是用来存储数据的。它们是为读取和写入单行数据而优化的,这也造就 了它的许多设计决策,包括它按行存储的结构。

在行存储或面向行的数据库中,数据逐行存储,因此行的第一列将在上一行的最后一列旁边。 例如,以下 Facebook\_Friends 数据:

#### Facebook\_Friends

Name	City	Age
Matt	Los Angeles	27
Dave	San Francisco	30
Tim	Oakland	33

这些数据将按行顺序存储在面向行的数据库中的磁盘上,如下所示:

Matt L	os Angeles	27 Dave	San Francisco	30 Tim	Oakland	33
--------	------------	---------	---------------	--------	---------	----

这样可以使数据库快速写一行,因为要写入该数据库,只需要将新行添加到数据末尾。

### 写入行存储数据库

让我们使用存储在数据库中的数据:

Matt	Los Angeles	27 Dave	San Francisco	30 Tim	Oakland	33
------	-------------	---------	---------------	--------	---------	----

#### 如果要添加新记录:

	0.0000000000000000000000000000000000000	
Jen	Vancouver	30

我们可以将其附加到当前数据的末尾:

Matt	Los Angeles	27 Dave	San Francisco	30	Tim	Oakland	33	Jen	Vancouver	30	
------	-------------	---------	---------------	----	-----	---------	----	-----	-----------	----	--

面向行的数据库仍然广泛用于联机事务处理(*OLTP*)样式的应用程序,因为它们可以很好地管理对数据库的写入。但是,数据库的另一个用例是分析其中的数据。这些联机事务处理(*OLAP*)用例需要一个能够支持数据临时查询(*ad hoc query*)的数据库。这是面向行的数据库比*C-store* 数据库慢的地方。

#### 从行存储数据库读取

面向行的数据库可以快速检索一行或一组行,但是执行聚合时,它将额外的数据(列)带入内存中,这比仅选择要聚合的列要慢。另外,面向行的数据库可能需要访问的磁盘数量通常更大。

#### 额外的数据存入内存

假设我们要从 Facebook\_Friends 数据中获取年龄总和。为此,我们将需要将所有这九段数据加载到内存中,然后提取相关数据进行汇总。

这浪费了计算时间。

#### 访问的磁盘数

假设一个磁盘只能容纳足够的字节数据,以便在每个磁盘上存储三列。在面向行的数据库中, 上表将存储为:

Disk 1					
Name	City	Age			
Matt	Los Angeles	27			

Disk 2					
Name	City	Age			
Dave	San Francisco	30			

Disk 3					
Name	City	Age			
Tim	Oakland	33			

为了获得所有年龄段的总和,计算机将需要浏览所有三个磁盘以及每个磁盘中的所有三列,以便进行此查询。

因此,我们可以看到,尽管将数据添加到面向行的数据库是快速简便的,但要从其中获取数据 可能需要使用额外的内存并需要访问多个磁盘。

### 列式数据库

创建数据仓库是为了支持分析数据。这些类型的数据库针对读取优化。

在 C-Store,列存储或面向列的数据库中,数据被存储为使得列的每一行将与同一列的其他行相邻。

让我们再次查看相同的数据集,并查看如何将其存储在面向列的数据库中。

#### Facebook\_Friends

Name	City	Age
Matt	Los Angeles	27
Dave	San Francisco	30
Tim	Oakland	33

#### 一张表按一行一行的顺序一次存储一列:

Matt	Dave	Tim	Los Angeles	San Francisco	Oakland	27	30	33
------	------	-----	-------------	---------------	---------	----	----	----

### 写入列存储数据库

如果要添加新记录:

Jen	Vancouver	30

我们必须在数据中导航,以将每列插入应位于的位置。

Matt	Dave	Tim	Jen	Los Angeles	San Francisco	Oakland	Vancouver	27	30	33	30	
------	------	-----	-----	-------------	---------------	---------	-----------	----	----	----	----	--

如果数据存储在单个磁盘上,它将与面向行的数据库存在相同的额外内存问题,因为它将需要 将所有内容都带到内存中。但是,面向列的数据库存储在单独的磁盘上时将具有明显的优势。

如果我们将上面的表放置在数据磁盘的类似限制的三列中,它们将像这样存储:

Disk 1						
Name						
Matt Dave Tim						

Disk 2						
City						
Los Angeles San Francisco Oakland						

Disk 3					
Age					
27	30	33			

#### 从列存储数据库中读取

要获得年龄的总和,计算机只需要访问一个磁盘(磁盘3)并求和其中的所有值。不需要引入额外的内存,它可以访问最少数量的磁盘。

尽管这例子有点过分简化,但它说明通过按列组织数据,将减少需要访问的磁盘数量,并将必 须保留在内存中的额外数据量降至最低。这大大提高了计算的整体速度。

面向列的数据库还有其他方法可以获得更高的性能。

### 将数据编码为更紧凑的形式

首先,让我们研究一种面向行或列的数据库可以使用的编码技术。下面例子将展示存储美国各州的列按字典和 bitmap 编码。

- 有 50 个,因此我们可以用 6 个位编码整个数据库,因为 6 个位提供 64 个唯一的模式。
- 存储实际的缩写将需要 16 位,因为需要提供两个 ASCII 字符中每个字符的 256 个唯一模式。
- 最糟糕的是,如果我们存储全名,则长度将是可变的,所需的位数会更多。

现在让我们看一下游程编码(run-length encoding,缩写 RLE)。这使您可以用计数(count)和值指示符(value indicator)替换相同值的任何序列。例如,我们可以将 aaaab 替换为 4a1b。当您创建带有已排序列的投影(projections)时,这将变得更加强大,因为所有相同的值都彼此相邻。

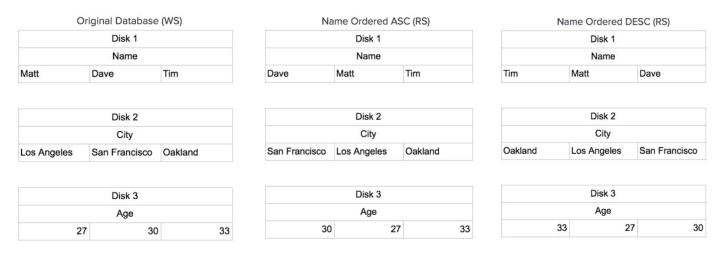
### 压缩数据

如果每条数据的位数相同,那么所有数据都可以进一步压缩为单条数据的位数乘以该位数。

#### 排序数据

进行数据临时查询(ad hoc query)时,会有多种排序可以提高性能。例如,我们可能希望按日期升序或降序。我们可能会在单个客户上寻找大量数据,因此按客户订购可以提高性能。

在面向行的数据库中,可以创建索引,但是数据很少以多种排序存储。但是,在面向列的数据库中,可以以任意多的方式存储数据。实际上,除了查询性能外,还有其他好处。这些不同排序的列称为投影(projections),它们使系统具有更高的容错能力,因为数据已存储多次。



要更新这些表看起来很复杂,而且确实如此。这就是为什么 C-store 数据库的架构具有可写存储(WS)和读取优化存储(RS)的原因。可写存储按照添加顺序对数据进行排序,以使向其中添加数据更加容易。我们可以轻松地将相关字段追加到我们的数据库中,如下所示:

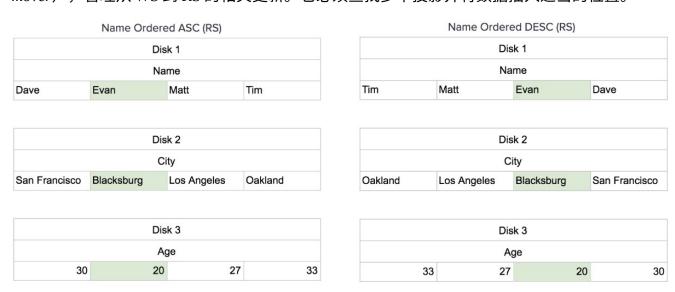
Original Database (WS)

	7.75		2550			
		Disk 1				
Name						
Matt	Dave	Tim	Evan			

Disk 2							
City							
Los Angeles San Francisco Oakland Blacksburg							

	Disk 3						
	Age						
27 30 33 20							

然后,经过读取优化的存储可以具有多个投影。然后,它具有一个元组移动器(tuple mover),管理从 WS 到 RS 的相关更新。它必须查找多个投影并将数据插入适当的位置。



这种架构意味着,当数据从WS更新到RS时,对RS的查询必须忽略部分添加的数据,直到更新完成。

## 摘要

面向列的数据库在 <u>2005 年发表了一篇论文,</u>解释了 Redshift,BigQuery 和 Snowflake 都是基于此设计的。这就是为什么它们都具有几乎相同的性能和相对成本的原因。大多数主要的云数据仓库提供商都使用这种面向列的数据库。这已成为关系数据库中支持 OLAP 的主要体系结构。

Written by: Blake Barnhill, Matt David