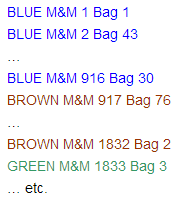
找出所有的红色M&Ms: 关于索引查询和全表扫描

*在这客座文章中,* [*Chris Saxon*](http://www.chrissaxon.co.uk/) *通过关于巧克力的分析说明一个十分重要的课题:什么时候数据库使用索引，什么时候不使用索引。虽然Chris的分析说明围绕着Oracle数据库展开，但这原理同样也在其他数据库中适用。*

当人们开始进行查询优化时，一个常见的问题就会出现，“为什么这查询不使用我想要的索引呢？”。这里有一些关于数据库优化使用索引的误解。其中一个我听说的是当查询表中5%及以下的行时，才会使用索引。但实际并非如此 – 使用索引与否的决定取决于它的使用成本。

数据库是如何影响使用索引的成本？

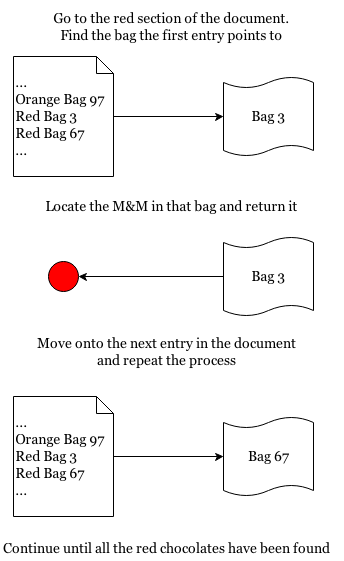
在了解详细细节之前，我们来聊聊巧克力！设想一下，你有100袋M&M’s。除此之外，你还有一个文档列表，其中包含着每个M&M的颜色和它所在袋的信息。该列表以颜色排序，所以我们首先有全部的蓝色(Blue)糖，其次棕色(Brown)，绿色(Green)等等。



现在需要找到所有的红色M&M’s。这里有一组基础的方法帮助你完成这次任务：

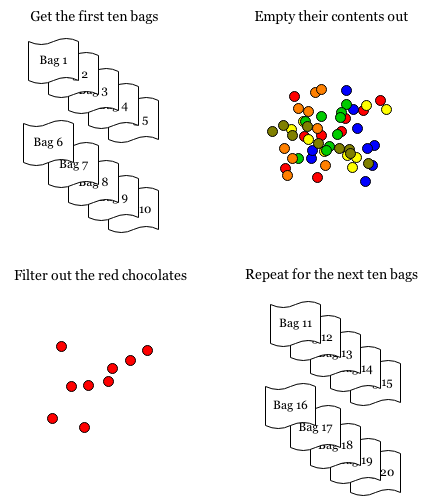
方法 1

通过文档列表列出每个M&M的颜色和位置。转到“红色”部分的顶端。查找第一个红色M&M的位置，拿取该位置的糖袋，得到那红色的M&M糖。接着返回至文档列表，重复刚刚的步骤从而获得下一个红色的M&M糖。不断地在文档列表和拿取糖袋中来回，直到我们查找完整个“红色”部分。



方法 2

我们一次性拿取若干袋（例如10个），清空里面的东西，拾取红色的M&M，然后将剩下的放回他们原来的袋中。



那么哪种方法更快呢？

直观来看，第二种方法似乎更快。我们只需要选择若干袋，然后进行一些筛选即可。反观第一种方法，我们需要进行许多文档和糖袋中的重复操作。这意味着我们必须对每个袋查看多次。

针对以上的判断，我们需要再严谨一些。计算一下当我们将所有的红色M&M拿取完时，两种方法各需要多少次操作。

当我们使用方法1时，每次我们通过文档列表找到一个新红色M&M的位置，并拿取该位置的糖袋视为一次操作。你有100个袋子，每个袋子里有55颗糖。这意味着你需要进行大约920次（100袋x55颗/6种颜色）操作（加上一些需要找到文档列表中“红色”部分的操作）。所以使用文档列表方法的操作次数在920次上下。

使用第二种方法，我们一次性可以收集10袋糖果。这意味着你需要进行10次（100袋/10袋/次=）操作（加上一些筛选巧克力的步骤），次数约为10次。

对比两种方法的操作次数（920 vs 10），显然第二种方法更具优势。

设想一下另一种场景。Mars计划进行一个促销活动，每100个袋子中含有一个银色M&M。如果你得到银色M&M，那么就会得奖。这种情况下，我们需要找到银色巧克力！

在这种情况下，使用方法1，我们通过文档列表找到那一颗银色糖的位置，接着拿取那袋糖，从而得到银色M&M。一次操作（也许是两次，包括用文档列表找到银色巧克力的位置），所以方法1 是两次操作。

接下来看看方法2，我们仍旧需要拾取每袋糖（和一些筛选）仅仅是为了找到那一颗银色糖果-操作次数固定在10次。显然，在Mars进行促销活动的情况下，方法1更胜一筹。

那么M&M巧克力和数据库有什么联系呢？

当Oracle将一行记录存储到数据库时，它将被放置在块中。就像许多M&Ms在同一袋中，（正常情况下）有许多行记录在同一块中。当需要访问某一特定的记录行时，Oracle将获取记录行所在的整个块，并从中检索出所请求的记录行。这就类似于我们拿起一袋M&Ms，接着挑选出其中的一块巧克力。

当进行索引范围扫描时，Oracle将搜索索引（文档列表）以查找与where字句匹配的第一个值。然后它将进行索引和数据表块的重复工作中，通过索引所指向的位置获取记录行。这类似于方法1，在文档列表和M&M糖袋中不断切换。

随着索引访问行数的增加，数据库需要完成更多的工作。因此，使用索引的成本会随着需要获取记录行数的增加而增加。

当进行全表扫描时，Oracle将在多数据块读取中提取若干块。然后过滤所提取的数据，以便只返回与where字句所匹配的行（本例中红色的M&Ms）- 其余的将被丢弃，就像方法2一样。

预期返回的行数对全表扫描工作的影响不大。它的主要成本取决于数据表的大小和每次获取的块数。

当匹配数据表中“较高”比率的数据行时，一次获取多数据块后进行筛选远比多次访问单个数据块高效得多。

什么时候索引扫描比全表扫描高效呢？

在上述M&M例子中，方法2（全表扫描）通过10次操作就能获取所有100袋。反观方法1（索引扫描），对于每颗糖都需要一次单独的操作。所以，在查找10颗M&M以内时，索引扫描比全表扫描高效。

Mars在每个袋中放了约55颗M&Ms，所以每个表（平均含有10颗M&M’s）的百分比为（10M&M’s / （100 bags \* 55 sweets）\*100 =）0.2%！

那么当Mars释放一些“巨大”M&M’s，仅仅有10颗糖在一个袋子中的时候呢？在这种情况下，总计来看糖更少了，所以等式中的分母减小了。我们全表扫描方法的成本固定在100袋需要10次操作。这意味着大约查询1%（10/（10\*100）\*100）的记录行时，索引查询表现更好。看起来上升的百分比，实际上仍是很小。

如果他们释放“迷你”M&M’s，每袋中含有200颗糖，分母会增大。这意味着当查询一个非常小占比的记录行时，索引查询更加高效。

所以当我们增加空间以存储记录行时，索引查询会比全表扫描高效。索引查询记录行的次数是固定不变的。然而，数据块因存储数据的需要而增加，这使得使用全表扫描的成本会上升许多。

然而，上述的推理有个巨大的假设条件。M&M’s的顺序与袋子不存在相关性。例如，第一颗红色M&M可能在袋1，第二颗在袋56，第三颗在袋20等。

现在设定一个不同的假设 – 文档列表中红色巧克力豆的顺序与他们所在袋子存在相关性。所以，第一批9个红色糖在袋1，下一批9个红色糖在袋2等。当我们仍需要访问所有100袋，我们可以保留最后一袋在手里，仅仅每隔9袋就更换一次。这样就能减少操作的次数，使得索引查询更加高效。

我们可以考虑得更进一步。如果Mars变更了装袋过程使得每个袋中只有一种颜色呢？

现在，代替访问每个袋子以得到所有红色糖的方法，我们只需要查询约16袋（100袋/6颜色/袋）即可。如果糖是和列表文档中同样的顺序进行放置（那么M&M’s 1-55全是蓝色且在袋1，袋2有蓝色M&M’S 56-100，以此类推到袋100，黄色M&M’s是496-550）,我们将会得到不更换袋子和更少袋子查找的好处。

这个原理 — 表中记录行的顺序与相应索引顺序的匹配程度 — 被称为集群因子。当数据表中物理顺序与索引相同（袋中的所有糖是同一颜色），数字越低；当两者相关性很小时，数字越大。

当集群因子与数据表中的块数越接近，索引查询越有可能被使用（分配的成本越低）；当它与数据表的行数接近时，越有可能使用全表扫描（索引访问的成本越高）。

总结一下

总而言之，我们可以通过基于成本考虑的优化以决定使用索引查询还是全表扫描：

* 获取数据表的块数，并除以多数据块读取数据块的个数得出全表扫描的成本。
* 对于查询语句可使用的每个数据表索引：
  + 计算出一个查询语句中数据表期望数据行的百分比（选择率）
  + 计算的百分比用于确定预期访问索引的百分比
  + 选择率乘以集群因子估算出通过索引匹配记录行的数据表块个数
  + 将这些值相加以得到索引扫描的成本
* 将全表扫描的成本与索引扫描中最小的成本进行比较。

这仅仅是基本成本优化工作（Orcale）的概述。如果你想了解优化程序公式，需要阅读[Cost-Based Oracle Fundamentals](https://www.amazon.co.uk/Cost-Based-Oracle-Fundamentals-Experts-Voice/dp/1590596366/ref=sr_1_1?ie=UTF8&qid=1502472304&sr=8-1&keywords=Cost-Based+Oracle+Fundamentals)书。[Jonathan Lewis](https://jonathanlewis.wordpress.com/), [Richard Foote](https://richardfoote.wordpress.com/) 和Markus 网站博客上的文章也包含了许多关于此主题，内容细节的帖子。

关于作者



Chris Saxon是Oracle开发人员倡导者。他的职责是帮助你充分使用Oracle数据库和享受SQL带来的乐趣。

为了更好地帮助您，[All Things SQL](https://blogs.oracle.com/sql/) 上有他的博客，[The Magic of SQL](https://www.youtube.com/c/TheMagicofSQL)有他创作的视频。