

## 25 | Hash索引的底层原理是什么？

2019-08-07 陈旸

SQL必知必会

[进入课程 >](#)



**讲述：陈旸**

时长 06:49 大小 6.26M



我们上节课讲解了 B+ 树的原理，今天我们来学习下 Hash 的原理和使用。Hash 本身是一个函数，又被称为散列函数，它可以帮助我们大幅提升检索数据的效率。打个比方，Hash 就好像一个智能前台，你只要告诉它想要查找的人的姓名，它就会告诉你那个人坐在哪个位置，只需要一次交互就可以完成查找，效率非常高。大名鼎鼎的 MD5 就是 Hash 函数的一种。

Hash 算法是通过某种确定性的算法（比如 MD5、SHA1、SHA2、SHA3）将输入转变为输出。相同的输入永远可以得到相同的输出，假设输入内容有微小偏差，在输出中通常会有不同的结果。如果你想要验证两个文件是否相同，那么你不需把两份文件直接拿来比对，只需要让对方把 Hash 函数计算得到的结果告诉你即可，然后在本地同样对文件进行 Hash 函数的运算，最后通过比较这两个 Hash 函数的结果是否相同，就可以知道这两个文件是否相同。

Hash 可以高效地帮我们完成验证的工作，它在数据库中有广泛的应用。今天的课程主要包括下面几个部分：

1. 动手写程序统计一下 Hash 检索的效率。
2. 了解 MySQL 中的 Hash 索引，理解使用它的优点和不足。
3. Hash 索引和 B+ 树索引的区别以及使用场景。


## 动手统计 Hash 检索效率

我们知道 Python 的数据结构中有数组和字典两种，其中数组检索数据类似于全表扫描，需要对整个数组的内容进行检索；而字典是由 Hash 表实现的，存储的是 key-value 值，对于数据检索来说效率非常快。

对于 Hash 的检索效率，我们来个更直观的认知。下面我们分别看一下采用数组检索数据和采用字典（Hash）检索数据的效率到底有怎样的差别。

实验 1：在数组中添加 10000 个元素，然后分别对这 10000 个元素进行检索，最后统计检索的时间。

代码如下：


 复制代码

```
1 import time
2 # 插入数据
3 result = []
4 for i in range(10000):
5     result.append(i)
6 # 检索数据
7 time_start=time.time()
8 for i in range(10000):
9     temp = result.index(i)
10 time_end=time.time()
11 print('检索时间', time_end-time_start)
```

运行结果：

检索时间为 1.2436728477478027 秒

实验 2：采用 Hash 表的形式存储数据，即在 Python 中采用字典方式添加 10000 个元素，然后检索这 10000 个数据，最后再统计一下时间。代码如下：

 复制代码

```
1 import time
2 # 插入数据
3 result = {}
4 for i in range(1000000):
5     result[i] = i
6 # 检索数据
7 time_start=time.time()
8 for i in range(10000):
9     temp = result[i]
10 time_end=time.time()
11 print('检索时间: ',time_end-time_start)
```

运行结果：

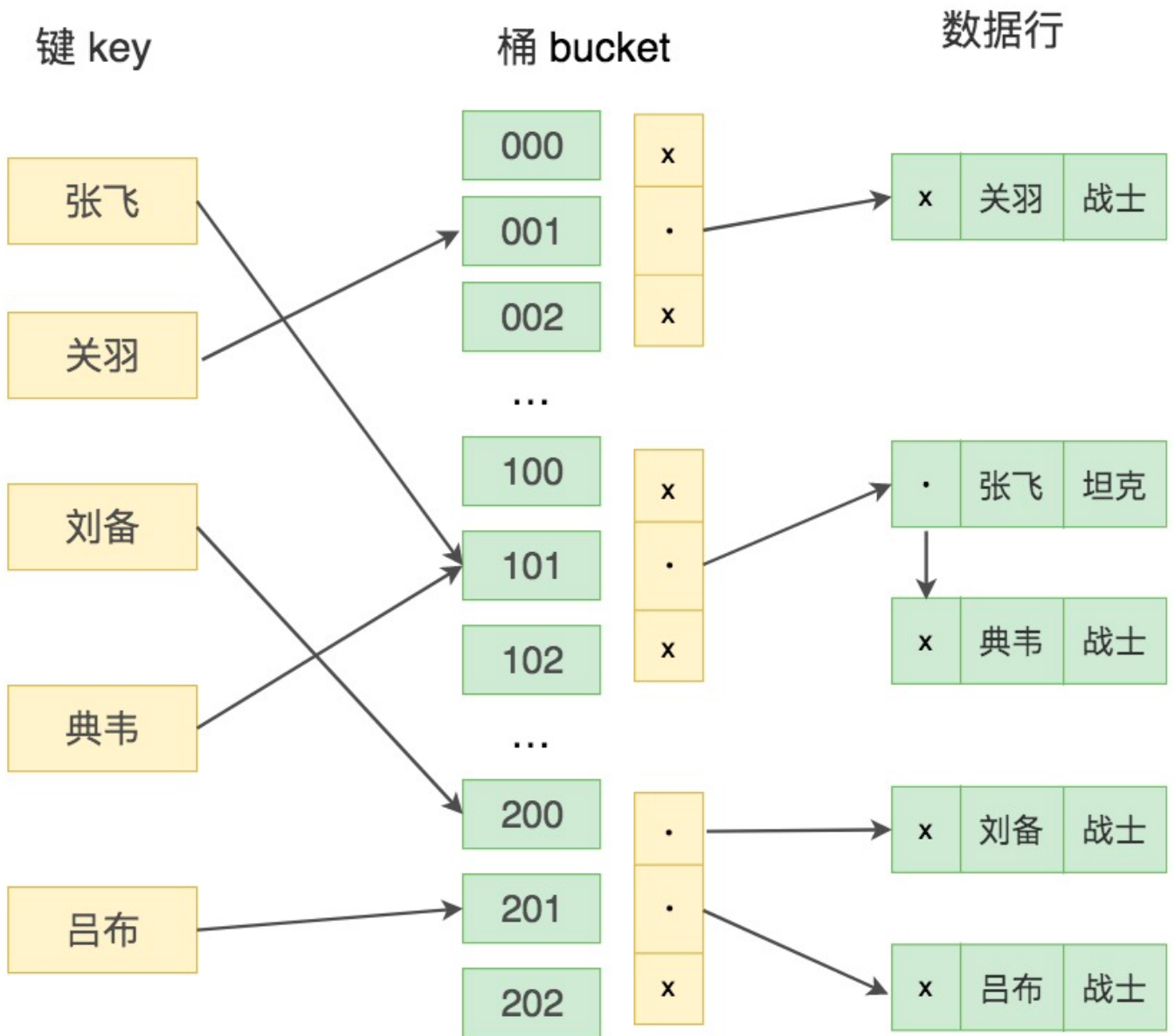
检索时间为 0.0019941329956054688 秒。

你能看到 Hash 方式检索差不多用了 2 毫秒的时间，检索效率提升得非常明显。这是因为 Hash 只需要一步就可以找到对应的取值，算法复杂度为  $O(1)$ ，而数组检索数据的算法复杂度为  $O(n)$ 。

## MySQL 中的 Hash 索引

采用 Hash 进行检索效率非常高，基本上一次检索就可以找到数据，而 B+ 树需要自顶向下依次查找，多次访问节点才能找到数据，中间需要多次 I/O 操作，从效率来说 Hash 比 B+ 树更快。

我们来看下 Hash 索引的示意图：



键值 key 通过 Hash 映射找到桶 bucket。在这里桶 ( bucket ) 指的是一个能存储一条或多条记录的存储单位。一个桶的结构包含了一个内存指针数组，桶中的每行数据都会指向下一行，形成链表结构，当遇到 Hash 冲突时，会在桶中进行键值的查找。

那么什么是 Hash 冲突呢？

如果桶的空间小于输入的空间，不同的输入可能会映射到同一个桶中，这时就会产生 Hash 冲突，如果 Hash 冲突的量很大，就会影响读取的性能。

通常 Hash 值的字节数比较少，简单的 4 个字节就够了。在 Hash 值相同的情况下，就会进一步比较桶 ( Bucket ) 中的键值，从而找到最终的数据行。

Hash 值的字节数多的话可以是 16 位、32 位等，比如采用 MD5 函数就可以得到一个 16 位或者 32 位的数值，32 位的 MD5 已经足够安全，重复率非常低。

我们模拟一下 Hash 索引。关键字如下所示，每个字母的内部编码为字母的序号，比如 A 为 01，Y 为 25。我们统计内部编码平方的第 8-11 位（从前向后）作为 Hash 值：

关键字	内部编码	内部编码平方	Hash值
ABCD	01020304	001041020252416	2025
ABCE	01020305	001041022293025	2229
.....	.....	.....	.....
YYAB	25250102	637567651010404	5101

## Hash 索引与 B+ 树索引的区别

我们之前讲到过 B+ 树索引的结构，Hash 索引结构和 B+ 树的不同，因此在索引使用上也会有一些差别。

1. Hash 索引不能进行范围查询，而 B+ 树可以。这是因为 Hash 索引指向的数据是无序的，而 B+ 树的叶子节点是个有序的链表。
2. Hash 索引不支持联合索引的最左侧原则（即联合索引的部分索引无法使用），而 B+ 树可以。对于联合索引来说，Hash 索引在计算 Hash 值的时候是将索引键合并后再一起计算 Hash 值，所以不会针对每个索引单独计算 Hash 值。因此如果用到联合索引的一个或者几个索引时，联合索引无法被利用。
3. Hash 索引不支持 ORDER BY 排序，因为 Hash 索引指向的数据是无序的，因此无法起到排序优化的作用，而 B+ 树索引数据是有序的，可以起到对该字段 ORDER BY 排序优化的作用。同理，我们也无法用 Hash 索引进行模糊查询，而 B+ 树使用 LIKE 进行模糊查询的时候，LIKE 后面模糊查询（比如 % 开头）的话就可以起到优化作用。

对于等值查询来说，通常 Hash 索引的效率更高，不过也存在一种情况，就是索引列的重复值如果很多，效率就会降低。这是因为遇到 Hash 冲突时，需要遍历桶中的行指针来进行比较，找到查询的关键字，非常耗时。所以，Hash 索引通常不会用到重复值多的列上，比如列为性别、年龄的情况等。

## 总结

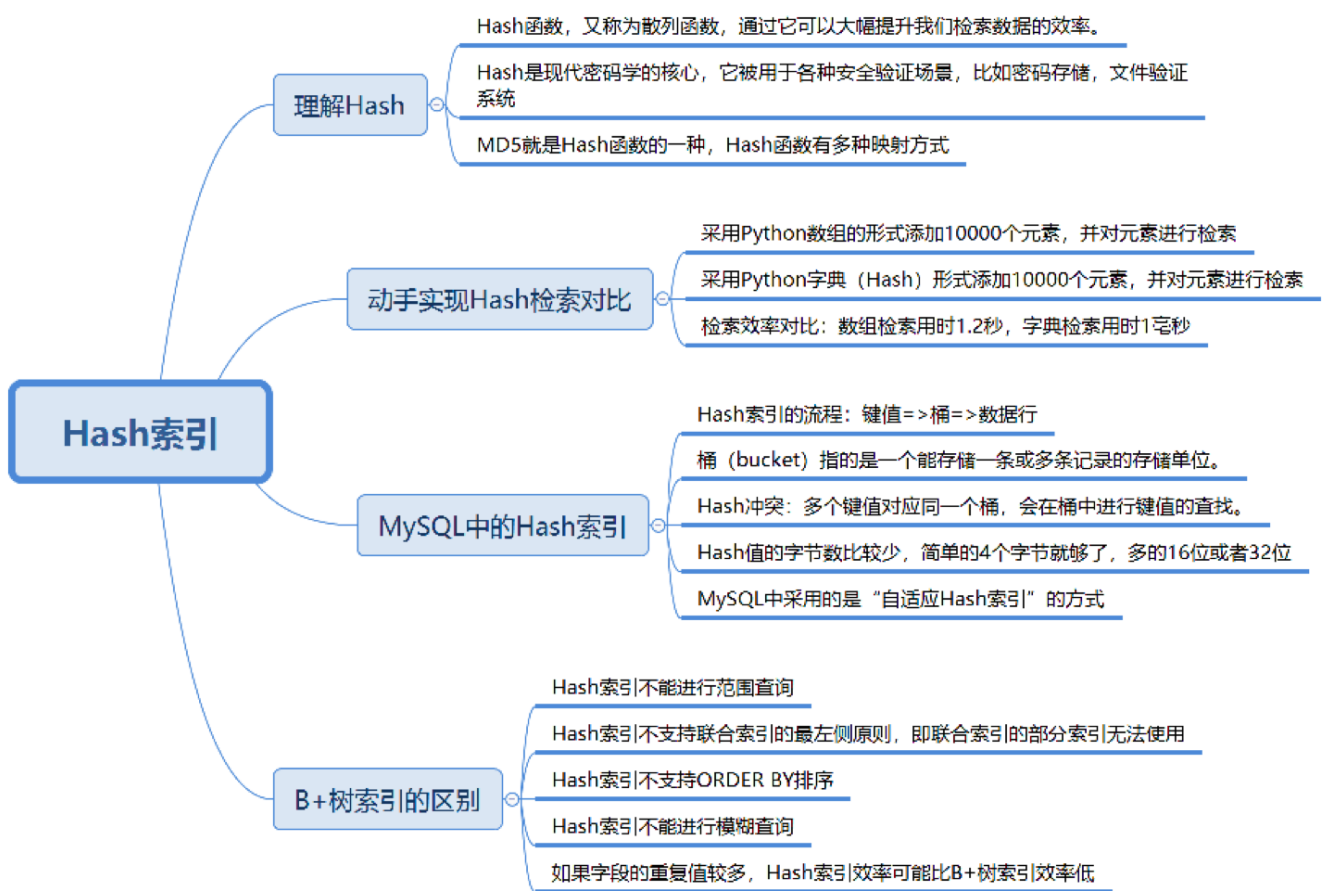
我今天讲了 Hash 索引的底层原理，你能看到 Hash 索引存在着很多限制，相比之下在数据库中 B+ 树索引的使用面会更广，不过也有一些场景采用 Hash 索引效率更高，比如在键值



型 ( Key-Value ) 数据库中，Redis 存储的核心就是 Hash 表。

另外 MySQL 中的 Memory 存储引擎支持 Hash 存储，如果我们需要用到查询的临时表时，就可以选择 Memory 存储引擎，把某个字段设置为 Hash 索引，比如字符串类型的字段，进行 Hash 计算之后长度可以缩短到几个字节。当字段的重复度低，而且经常需要进行等值查询的时候，采用 Hash 索引是个不错的选择。

另外 MySQL 的 InnoDB 存储引擎还有个“自适应 Hash 索引”的功能，就是当某个索引值使用非常频繁的时候，它会在 B+ 树索引的基础上再创建一个 Hash 索引，这样让 B+ 树也具备了 Hash 索引的优点。



今天的内容到这里就结束了，我留两道思考题吧。查找某个固定值时 Hash 索引比 B+ 树更快，为什么 MySQL 还要采用 B+ 树的存储索引呢？另外，当两个关键字的 Hash 值相同时会发生什么？

欢迎你在评论区写下你的思考，我会和你一起交流，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。


# SQL 必知必会

从入门到数据实战

陈旻

清华大学计算机博士



新版升级：点击「请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 24 | 索引的原理：我们为什么用B+树来做索引？

下一篇 26 | 索引的使用原则：如何通过索引让SQL查询效率最大化？

## 精选留言 (13)

 写留言



蒙开强

2019-08-07

老师，你好，hash索引与B+树索引是在建索引的时候手动指定么



2



用0和1改变自己

2019-08-08

1，Hash索引有很大的限制，如联合索引、模糊查询、范围查询，以及列里有重复值多。  
2，需要遍历链表中所有行指针，逐一进行比较，直到找到所有符合条件的

作者回复: 对的



1



**许童童**

2019-08-07

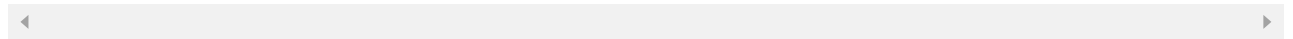
查找某个固定值时 Hash 索引比 B+ 树更快，为什么 MySQL 还要采用 B+ 树的存储索引呢？另外，当两个关键字的 Hash 值相同时会发生什么？

因为B+ 树的一些特性像范围查询，联合索引的最左侧原则，支持 ORDER BY 排序等Hash索引没有。

会发生Hash冲突，然后去按key顺序在桶中等值查找。

展开 ▾

作者回复: 对的



1



**我行我素**

2019-08-07

回复下蒙开强，如果是使用navicat创建索引的时候在后面是可以直接选择索引类型的，如果使用sql创建索引就是在穿件的使用using指定，一般默认是B+

展开 ▾



1



**马哲富**

2019-08-12

老师您好，是不同的索引结构对应不同类型的索引（比如聚集索引、非聚集索引等）吗？另外知道这些底层的索引结构对于一个普通的开发人员的价值点（或者说判断依据）在哪儿呢？



**渴望飞的哺乳类**

2019-08-11

老师，B+ 树使用 LIKE 进行模糊查询的时候，like 'xx%' 才会使用到索引吧



**wusiration**

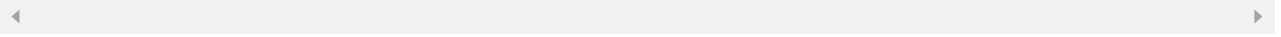
2019-08-08

mysql查询中存在着很多范围查询、order by的场景，在这些场景下，B+树的性能好于Hash索引；关键字出现相同Hash码时，会出现hash冲突。

展开 ▾



作者回复: 对的 所以对于一般需求来说, B+树在数据库应用的场景更多, Hash适用一些特殊的需求, 比如文件校验, 密码学等

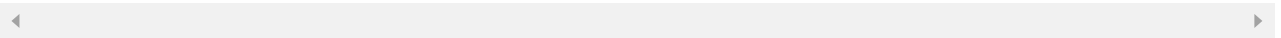


**ABC**

2019-08-08

感觉Hash索引和Java的HashMap的Hash实现有点像, 不过Java用链地址法解决了Hash冲突的问题。

作者回复: 对 原理上是一样的

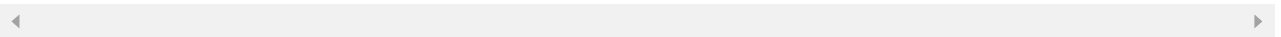


**Ashlar**

2019-08-08

能不能请老师分别推荐一下学习MySQL, Oracle, sql Server的一些书籍或者资料呢?

作者回复: 可以看下关于MySQL高性能优化的书籍, 如果是数据库初学者也可以先从SQL Server开始, 毕竟微软的产品在操作界面上上手简单。书籍有《21天学通SQL Server》《SQL优化最佳实践》《MySQL技术内容: SQL编程》《Oracle从入门到精通》



**一语中的**

2019-08-08

来自信息安全专业, 看到这一节hash索引原理中提到hash算法, hash是不可逆的, 有种异常熟悉的感觉, 嗯, 那些年学的安全算法们AES, DES, IDEA, Hash, HMAC...

展开 ▾

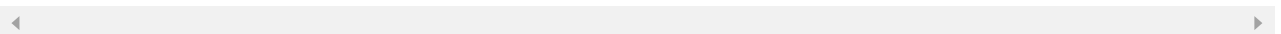


**Geek\_Wison**

2019-08-07

前模糊查询具体指啥, 能举一个具体的例子吗? 比如是指: 'a%' 还是指 '%a' ?

作者回复: 前模糊查询就是类似 %a 这种, 因为在字符串匹配的前面就是模糊查询了





许童童

2019-08-07

老师你好，数组检索数据的算法复杂度为  $O(n)$ 。  
不应该也是 $O(1)$ 吗？

作者回复: 感谢提问，一个数组如果有 $n$ 个元素，需要遍历完所有的元素才能找到某个元素，所以是 $O(n)$ ，如果是 $O(1)$ 就是不需要遍历，直接找到那个元素



2



Destroy

2019-08-07

有个疑问，在数组中，针对下标的检索，时间复杂度是 $O(1)$ 。老师的代码中用的是`result.index(i)`，这个函数用的应该不是下标检索。因为当我把代码改成`result[i]`，检索时间 0.0009975433349609375

作者回复: 因为我们要找的是某个元素的值，比如我添加的元素是1, 3,5,7...99 一共50个元素，如果我想要找7这个元素，你会用7作为下标进行检索，还是将7作为元素值进行查找呢？这里就需要检索具体的数值，对于数组来说下标是自动分配的，所以我们需要遍历数组来找到某个数值。

而对于字典来说，我们就可以创建索引了



2

