25丨Hash索引的底层原理是什么？

2019-08-07 陈旸

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，那么你不需要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：

```
import time
# 插入数据
result = []
for i in range(10000):
    result.append(i)
# 检索数据
time_start=time.time()
for i in range(10000):
    temp = result.index(i)
time_end=time.time()
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：

```
import time
# 插入数据
result = {}
for i in range(1000000):
    result[i] = i
# 检索数据
time_start=time.time()
for i in range(10000):
    temp = result[i]
time_end=time.time()
print('检索时间：',time_end-time_start)
```

运行结果：

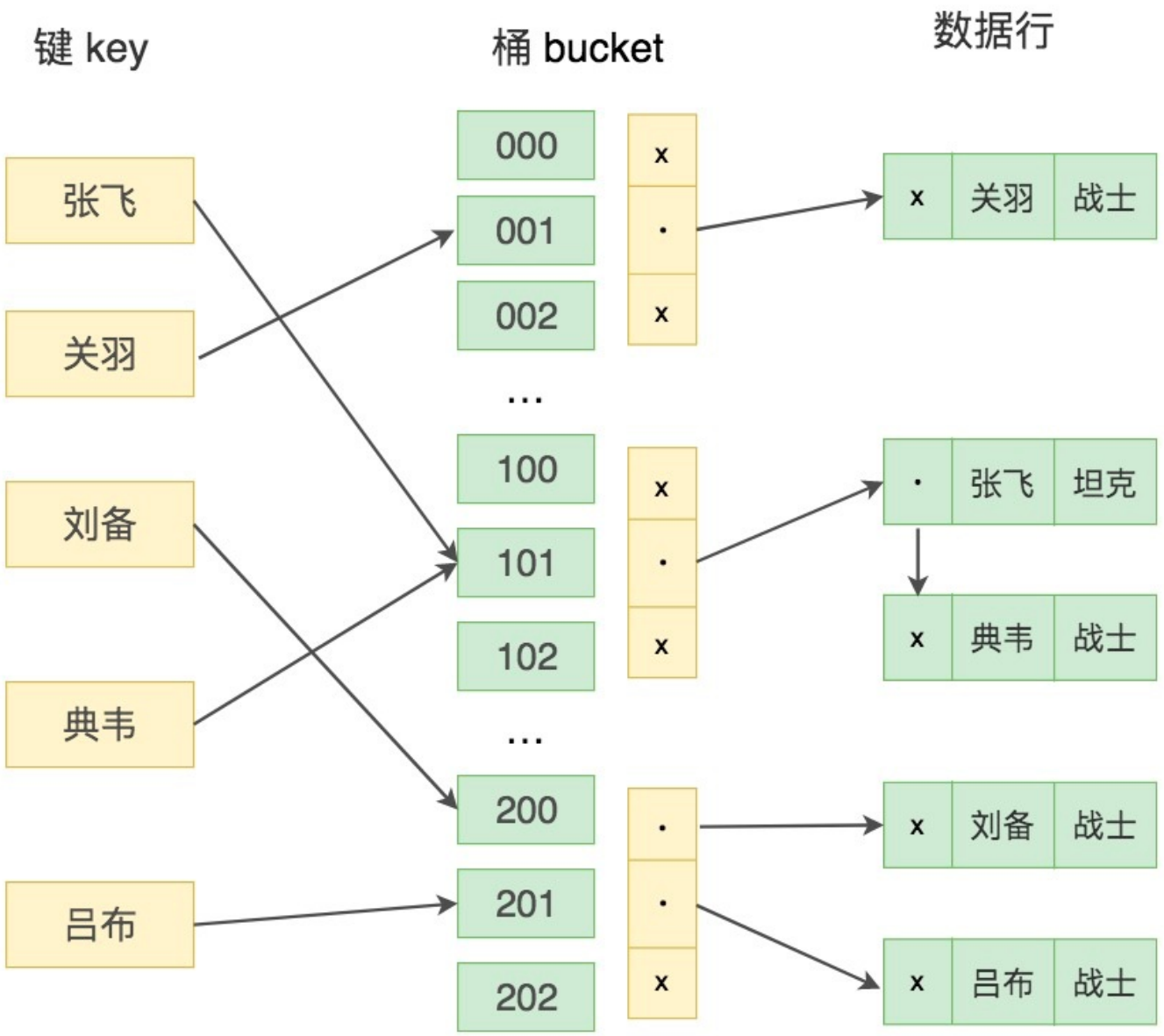
检索时间为0.0019941329956054688秒。

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后面前模糊查询（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。

精选留言

用0和1改变自己 3

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

2019-08-08

作者回复

对的

2019-08-09

我行我素 3

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+

2019-08-07

蒙开强 2

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

2019-08-07

wusiration 1

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。

2019-08-08

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

2019-08-09

许童童 1

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。

2019-08-07

作者回复

对的

2019-08-09

Destroy、 1

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

2019-08-07

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

2019-08-09

小智e 0

所以老师能在稍微解释一下“自适应 Hash 索引”吗？自己查了一些资料，不是很懂。

2019-08-22

Geek_1c165d 0

老师有两个问题：

1、是不是创建的索引，不管是Hash索引还是B树索引都会存储在硬盘上的么？

2、B树索引的内容以B树的数据结构进行存储，那Hash索引是以什么数据结构进行存储的？

2019-08-14

马哲富 0

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？

2019-08-12

渴望飞的哺乳类

0

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

2019-08-11

ABC

0

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。

2019-08-08

作者回复

对 原理上是一样的

2019-08-09

Ashlar

0

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？

2019-08-08

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》

2019-08-09

一语中的

0

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...

2019-08-08

Geek_Wison

0

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' ？

2019-08-07

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

2019-08-09

许童童

0

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？

2019-08-07

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

2019-08-09