

**大数据存储系统与管理课程报告**

|  |  |
| --- | --- |
| 姓 名： | 刘少东 |
| 学 院： | 计算机科学与技术学院 |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS2102 |
| 学 号： | U202190067 |
| 指导教师： | 华宇 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

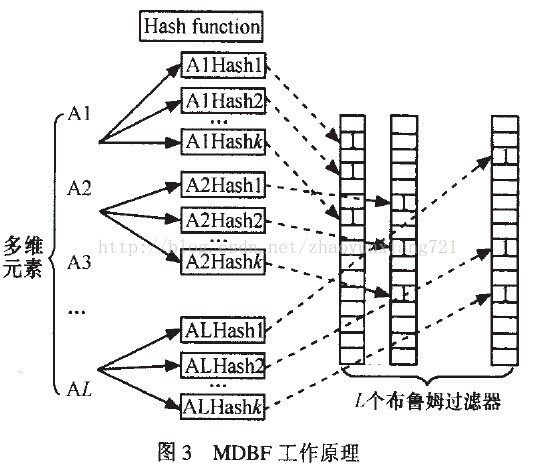
2024年 4 月 22 日

# 实验背景

多维布鲁姆过滤器（MDBF）是布鲁姆过滤器（Bloom Filter）的扩展，它允许在多个维度上有效地表示和查询数据。布鲁姆过滤器是一种空间效率极高的数据结构，用于测试一个元素是否是一个集合的成员，而MDBF则针对具有多个属性的数据项进行了优化。

MDBF通过为数据集中的每个维度使用独立的布鲁姆过滤器来工作。例如，如果数据集包含颜色、大小和形状三个维度，MDBF将为每个维度创建一个布鲁姆过滤器。当一个数据项被添加到MDBF时，它的每个属性都会被映射到相应维度的布鲁姆过滤器中。

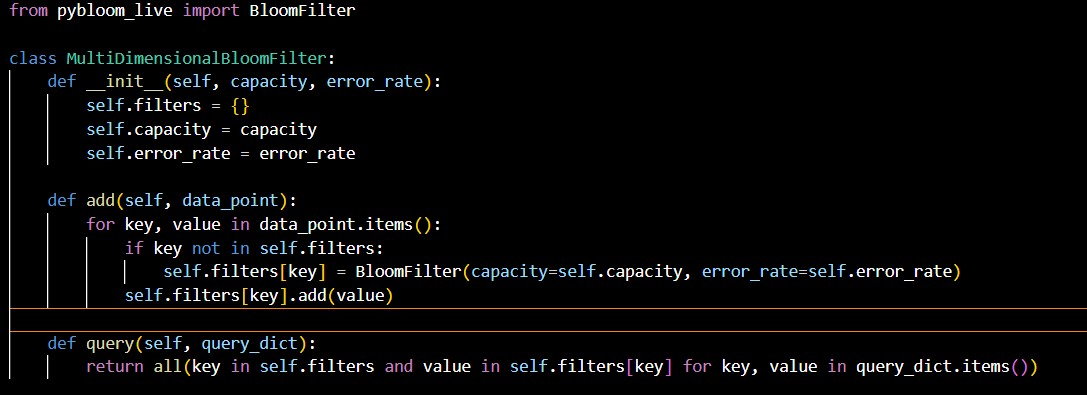
查询时，MDBF可以同时在多个维度上进行，这通过检查每个相关维度的布鲁姆过滤器来完成。只有当所有相关维度的布鲁姆过滤器都表明属性可能存在时，MDBF才会报告数据项可能存在。



# 二、布隆过滤器设计

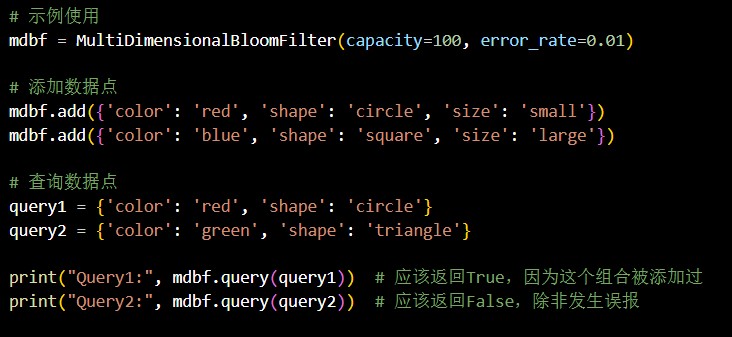
定义一个布隆过滤器

图 1 布隆过滤器数据结构



增加数据点，测试布隆过滤器

图 2 数据点



在这个示例中，我们定义了一个MultiDimensionalBloomFilter类，它包含了一个字典filters，该字典的键是属性名，值是对应属性的Bloom Filter。add方法用于向MDBF中添加数据点，它会为每个属性创建或更新一个Bloom Filter。query方法用于查询一个数据点是否可能存在于MDBF中，它会检查查询中的每个属性是否都在对应的Bloom Filter中。

由于Bloom Filter的特性，如果query方法返回True，这并不保证查询的数据点一定存在于MDBF中，只能说它可能存在（可能发生误报）。如果返回False，则可以确定查询的数据点一定不存在于MDBF中。

# 三、理论分析

布隆过滤器不存在false negative问题

而对于false positive问题，根源来自于hash冲突，因此可通过增加位数组大小、采用更均匀的hash函数、降低存储量等方式缓解。

# 性能分析

Bloom Filter的空间效率：Bloom Filter是一种空间效率极高的数据结构，它使用比传统数据结构（如哈希表或二叉树）少得多的空间来表示集合。在MultiDimensionalBloomFilter中，对于每个维度都创建了一个独立的Bloom Filter，这意味着空间开销随着维度的增加而线性增加。如果每个维度的数据点分布较为均匀，那么这种设计可以保持较高的空间效率。

动态创建Bloom Filter：在add方法中，如果遇到一个新的维度（即该维度的Bloom Filter尚未创建），则会动态创建一个新的Bloom Filter。这种动态创建策略意味着空间使用是按需分配的，有助于减少未使用空间的浪费。

添加操作（add方法）：每个数据点的添加操作涉及到对每个属性的遍历，以及可能的Bloom Filter创建和哈希计算。由于Bloom Filter的哈希计算通常很快，所以添加操作的时间复杂度主要取决于数据点的属性数量。对于具有固定数量属性的数据点，这个操作的时间复杂度是常数级别的。

查询操作（query方法）：查询操作的时间效率同样取决于查询字典中的键值对数量。由于Bloom Filter的查询操作是非常快速的（通常是常数时间），整个查询操作的时间复杂度也是常数级别的，前提是查询条件的数量固定。

Bloom Filter的误报率：Bloom Filter具有一定的误报率（false positive rate），即它可能错误地报告一个不在集合中的元素为存在。MultiDimensionalBloomFilter中的每个Bloom Filter都有自己的误报率，这个误报率在初始化时通过error\_rate参数指定。整个MultiDimensionalBloomFilter的误报率取决于各个维度Bloom Filter误报率的组合。

减少误报率：要减少整个MultiDimensionalBloomFilter的误报率，可以通过减少每个维度Bloom Filter的误报率来实现，例如通过增加Bloom Filter的大小或使用更多的哈希函数。然而，这会以增加空间开销为代价。