BPlusTree

测试评估报告

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <16/05/2020> | <1.0> | 测试完成后，撰写测试报告 | 赵胜龙 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

1.5 概述 4

2. 测试结果摘要 4

3. 基于需求的测试覆盖 4

4. 基于代码的测试覆盖 5

5. 缺陷报告 6

6. 测试结论与建议 6

测试评估报告

# 简介

本测试报告是在进行B+树的结构测试之后的总结报告，报告中主要对本次的测试结果进行汇总，以查阅分析测试效果和报告缺陷异常。

## 目的

B+树测试报告的目的如下：

* 汇总本次结构测试的结果，对缺陷进行分析
* 为以后的测试工作提供指导和记录。

## 范围

B+树测试报告建立在B+树源代码、B+树测试计划、B+树测试用例之上，文档中不会对代码进行分析，也不会对测试用例进行赘述，如果有需要可以查阅相关文档。

## 定义、首字母缩写词和缩略语

路径分析：对代码结构进行DD-路径的路径分析，以更清楚的展示代码结构，从而做到分支/条件/路径覆盖。

数据流分析：对代码中的数据定义、数据使用进行分析，区分出数据定义使用路径，定义清除路径，从而对测试进行更好的指导。

## 参考资料

《B+树测试用例》

《B+树测试计划》

《B+树路径分析及结构测试覆盖率报告》

《B+树数据流分析及数据流测试覆盖率报告》

## 概述

# 测试结果摘要

测试总共用了两种方法：基于路径的结构测试 以及 基于数据流的结构测试

在基于路径的结构测试中，覆盖率达到了100%，具体的覆盖率如下

Class: 100%（5/5） Method: 100%（38/38） Line: 100%（175/175）

在基于数据流的结构测试中，覆盖率接近100%，具体的覆盖率如下：

Class: 100%（5/5） Method: 100%（38/38） Line：97%（171/175）

所有的测试用例中，通过了73个测试用例，2个测试用例未通过，是B+树实现的bug导致。

# 基于需求的测试覆盖

本次B+树的测试中不针对需求进行测试，更多的是针对代码进行分析测试，因此本节不是本文档的重点。

# 基于代码的测试覆盖

* 1. 代码覆盖率及缺陷分布

**表1 需求覆盖率及缺陷分布**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试内容** | | **测试**  **用例数** | **需求**  **覆盖率** | **缺陷数** | **缺陷率** | **备注** |
| **方法接口** | BPlusTree | 1 | 100% | 0 | 0% |  |
| BPlusTree (带参数) | 2 | 100% | 0 | 0% |  |
| BPlusTree.search | 4 | 100% | 0 | 0% |  |
| BPlusTree.searchRange | 4 | 100% | 0 | 0% |  |
| BPlusTree.insert | 3 | 100% | 0 | 0% |  |
| BPlusTree.delete | 3 | 100% | 1 | 33.3% | 严重错误 |
| BPlusTree.toString | 4 | 100% | 0 | 0% |  |
| InternalNode.getValue | 2 | 100% | 0 | 0% |  |
| InternalNode.deleteValue | 3 | 100% | 0 | 0% |  |
| InternalNode.insertValue | 2 | 100% | 0 | 0% |  |
| InternalNode.getFirstLeafKey | 2 | 100% | 0 | 0% |  |
| InternalNode.getRange | 3 | 100% | 0 | 0% |  |
| InternalNode.merge | 1 | 100% | 0 | 0% |  |
| InternalNode.split | 1 | 100% | 0 | 0% |  |
| InternalNode.isOverflow | 2 | 100% | 0 | 0% |  |
| InternalNode.isUnderflow | 2 | 100% | 0 | 0% |  |
| InternalNode.getChild | 1 | 100% | 0 | 0% |  |
| InternalNode.deleteChild | 2 | 100% | 0 | 0% |  |
| InternalNode.insertChild | 2 | 100% | 0 | 0% |  |
| InternalNode. getChildLeftSibling | 2 | 100% | 0 | 0% |  |
| InternalNode. getChildRightSibling | 2 | 100% | 0 | 0% |  |
| LeafNode.LeafNode | 1 | 100% | 0 | 0% |  |
| LeafNode.getValue | 2 | 100% | 0 | 0% |  |
| LeafNode.deleteValue | 2 | 100% | 0 | 0% |  |
| LeafNode.insertValue | 2 | 100% | 0 | 0% |  |
| LeafNode.getFirstLeafKey | 2 | 100% | 1 | 50% | 错误不是很严重 |
| LeafNode.getRange | 4 | 100% | 0 | 0% |  |
| LeafNode.merge | 1 | 100% | 0 | 0% |  |
| LeafNode.split | 1 | 100% | 0 | 0% |  |
| LeafNode.isOverflow | 2 | 100% | 0 | 0% |  |
| LeafNode.isUnderflow | 2 | 100% | 0 | 0% |  |
|  | **数**据流 | 6 | 99% | 0 | 0% |  |
|  | **总计** | 75 | 100% | 0 | 2.7% |  |

* 1. 缺陷严重程度

# 缺陷报告

* 1. 缺陷清单

**表3 功能性缺陷列表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序**  **号** | **缺陷**  **编号** | **严重程度** | **接口模块** | **缺陷描述** |
| 1 | Bug001 | High | LeafNode.getFirstLeafKey | 当Leaf的key列表为空时，代码中直接return keys.get(0)，会造成越界错误 |
| 2 | Bug002 | Urgent | BPlusTree.delete | 在2层B+树情况下，比如  3  [1, 2], [3, 4]  删除3，会导致合并失败，破坏树的结构，测试中，书的结构会变成如下形态  3  [1, 2, 4], [4] |

# 测试结论与建议

本次测试表明，代码基本正确完整。但发现了仍存在的较严重缺陷，应当尽快进行修复。且此设计未考虑并发情况，尚不足以作为成熟的B+树项目进行应用。