Отчёт

**1)B-дерево** (по-русски произносится как **Би-дерево**) — [структура данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), дерево поиска. С точки зрения внешнего логического представления, [сбалансированное](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE), сильно ветвистое [дерево](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)). Часто используется для хранения данных во внешней памяти.

Использование B-деревьев впервые было предложено Р. Бэйером ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *R. Bayer*) и Э. МакКрейтом ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *E. McCreight*) в [1970 году](https://ru.wikipedia.org/wiki/1970_%D0%B3%D0%BE%D0%B4).

*Сбалансированность* означает, что длина любых двух путей от корня до листьев различается не более, чем на единицу.

*Ветвистость дерева* — это свойство каждого узла дерева ссылаться на большое число узлов-потомков.

2)С точки зрения физической организации B-дерево представляется как мультисписочная структура страниц памяти, то есть каждому узлу дерева соответствует блок памяти (страница). Внутренние и листовые страницы обычно имеют разную структуру. Время доступа к произвольному блоку на жёстком диске очень велико (порядка миллисекунд), поскольку оно определяется скоростью вращения диска и перемещения головок. Поэтому важно уменьшить количество узлов, просматриваемых при каждой операции. Использование поиска по [списку](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) каждый раз для нахождения случайного блока могло бы привести к чрезмерному количеству обращений к диску вследствие необходимости последовательного прохода по всем его элементам, предшествующим заданному, тогда как поиск в B-дереве, благодаря свойствам сбалансированности и высокой ветвистости, позволяет значительно сократить количество таких операций.

Особенности:

* Во всех случаях полезное использование пространства вторичной памяти составляет свыше 50 %. С ростом степени полезного использования памяти не происходит снижения качества обслуживания.
* Произвольный доступ к записи реализуется посредством малого количества подопераций (обращения к физическим блокам).
* В среднем достаточно эффективно реализуются операции включения и удаления записей; при этом сохраняется естественный порядок ключей с целью последовательной обработки, а также соответствующий баланс дерева для обеспечения быстрой произвольной выборки.
* Неизменная упорядоченность по ключу обеспечивает возможность эффективной пакетной обработки.

3) Поиск по B-дереву аналогичен поиску по двоичному дереву поиска. В двоичном дереве поиска поиск начинается с корня и каждый раз принимается двустороннее решение (пойти по левому поддереву или по правому). В В-дереве поиск также начинается с корневого узла, но на каждом шаге принимается n-стороннее решение, где n – это общее количество потомков рассматриваемого узла. В В-дереве сложность поиска составляет **O(log n)**.

5)

6) Данная структура данных хорошо работает с большими базами данных. Операция поиска намного эффективнее списочного поиска в жёстком диске, из-за использования оперативной памяти. Минусом является сложная по сравнению с бинарным деревом операция вставки, и ещё более сложная операция удаления (так как необходимо проверять правильность дерева и перестроить в случае нарушений). Данная СД используется для организации индексирования в современных СУБД.

7)

<https://habr.com/ru/post/114154/>

<https://habr.com/ru/company/otus/blog/459216/>

https://habr.com/ru/post/337594/

https://ru.wikipedia.org/wiki/B-дерево

Т. Кормен “Алгоритмы – построение и анализ”

Р. Сейджик ”Алгоритмы на Java”

4)

|  |  |  |  |
| --- | --- | --- | --- |
| **Поиск** | **Добавление** | **Удаление** | **Количество элементов** |
|  |  |  | 3000 |
| 0,792 | 0,0055 | 0,0108 |  |
| 0,2708 | 0,0068 | 0,0053 |  |
| 0,132 | 0,0018 | 0,0043 |  |
| 0,0281 | 0,0013 | 0,0041 |  |
| 0,1116 | 0,0014 | 0,0041 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Поиск** | **Добавление** | **Удаление** | **Количество элементов** |
|  |  |  | 10000 |
| 0,208 | 0,003 | 0,0053 |  |
| 0,1457 | 0,0022 | 0,0048 |  |
| 0,0276 | 0,0019 | 0,0061 |  |
| 0,0626 | 0,0021 | 0,0051 |  |
| 0,1206 | 0,0046 | 0,0041 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Поиск** | **Добавление** | **Удаление** | **Количество элементов** |
|  |  |  | 15000 |
| 0,0442 | 0,0012 | 0,0049 |  |
| 0,0207 | 0,0009 | 0,0168 |  |
| 0,124 | 0,0014 | 0,0047 |  |
| 0,1118 | 0,0013 | 0,0045 |  |
| 0,2299 | 0,0021 | 0,0248 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Поиск** | **Добавление** | **Удаление** | **Количество элементов** |
|  |  |  | 1000 |
| 0,1805 | 0,0021 | 0,0075 |  |
| 0,191 | 0,0019 | 0,0035 |  |
| 0,19 | 0,0016 | 0,0039 |  |
| 0,0347 | 0,001 | 0,0027 |  |
| 0,2241 | 0,0023 | 0,0046 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Поиск** | **Добавление** | **Удаление** | **Количество элементов** |
|  |  |  | 7000 |
| 0,1948 | 0,0023 | 0,0044 |  |
| 0,1252 | 0,0015 | 0,0039 |  |
| 0,0991 | 0,0012 | 0,016 |  |
| 0,051 | 0,0018 | 0,0036 |  |
| 0,0203 | 8,00E-04 | 0,0049 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Поиск** | **Добавление** | **Удаление** | **Количество элементов** |
|  |  |  | 5000 |
| 0,0436 | 0,0016 | 0,005 |  |
| 0,0301 | 0,001 | 0,0046 |  |
| 0,0255 | 9,00E-04 | 0,0046 |  |
| 0,0296 | 0,0012 | 0,0035 |  |
| 0,0245 | 0,001 | 0,0033 |  |

8)9)в репозитории