САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра системного программирования

Федотовский Павел Валерьевич

Влияние селективности на исполнение запросов в структурах многомерного индексирования

Дипломная работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Терехов А.Н.

Научный руководитель:
ассистент Чернышев Г.А.

Рецензент:
д. ф.-м. н., профессор Новиков Б.А.

Санкт-Петербург
2014

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Software Engineering Chair

Pavel Fedotovskii

Selectivity impact on query performance in multidimensional index structures.

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Andrey Terekhov

Scientific advisor:
assistant professor George Chernishev

Reviewer:
professor Boris Novikov

Saint-Petersburg
2014

Оглавление

1. Введение

Индексом называется избыточная структура хранения, предназначенная для ускорения выборки данных по некоторому классу условий на значения атрибутов. За счет особой организации данных, этот объект позволяет выполнять запрос существенно быстрее полного перебора записей, благодаря чему является одной из важных частей современной системы управления базами данных (СУБД).

Важным требованием при использовании структуры данных для организации индекса в СУБД является обеспечение корректности параллельного доступа. Это особенно важно для систем оперативной обработки транзакций (OLTP, on-line transactions processing) [27], которые имеют дело с большим количеством запросов, поступающих одновременно. Такие системы широко используются, например в банковском секторе, при автоматизации бронирования авиабилетов и т.д.

Следует обратить внимание на тот факт, что резкое снижение цен на оперативную память вызвало рост популярности индексов, полностью работающих в оперативной памяти. Существует мнение, что такие системы будут господствовать на рынке OLTP систем уже в скором времени [11]. В данной работе рассматриваются именно такие индексы.

Современные СУБД имеют специальные утилиты, которые по заданной нагрузке рекомендуют оптимальный набор индексов. Например, в СУБД Oracle для этого служит SQL Access Advisor, в Microsoft SQL Server - Database Engine Tuning Advisor [8]. Подобные утилиты есть и в других СУБД, таких как IBM DB2, PostgreSQL. Данная работа посвящена вопросу выбора оптимальной структуры данных для многомерного индекса, то есть в случае, когда поиск происходит по нескольким атрибутам.

Многомерные индексы применяются например, в геоинформационных системах (GIS) [13], в которых запрашиваются координаты объектов, таких как реки, города, озера, здания и др. В качестве индексируемых данных могут выступать многомерные точки или пространственные объекты. В данной работе рассматривается исключительно первый тип данных.

Наиболее популярным запросом к многомерному индексу является запрос на диапазон (range query) [25] — нахождение всех объектов, лежащих в данном многомерном прямоугольнике. Примером является поиск всех ресторанов в данной области карты

(задаваемой прямоугольником). Часто достаточно в качестве результата запроса выдать лишь первые k удовлетворяющих ему записей, где k - параметр [10]. При этом результирующие записи должны быть упорядочены по какому-то критерию. Примером такого запроса может служить нахождение самых населенных городов некоторой страны. Также условие выборки первых k записей может возникнуть в следующих случаях.

- При исполнении сложного запроса, оптимизатор запросов находит оптимальный план его исполнения. В процессе перебора вариантов, может возникнуть условие выборки первых k упорядоченных записей для ускорения исполнения последующих операций, например LEFT JOIN.

- При ручном просмотре результатов запроса, пользователю не нужен весь результат сразу, ему показывается только часть.

Нахождение оптимальной структуры данных является актуальной задачей, в частности на соревновании[1] при конференции ACM SIGMOD'13, была предложена задача выбора оптимальной структуры данных для индекса при рассмотренных выше ограничениях.

Важным параметром запроса, влияющим на время его исполнения, является селективность — процент записей от общего числа, которые являются результатом запроса. Формально, если общее количество записей равно n, а запросу q удовлетворяет k записей, то селективность запроса q равна $\frac{k}{n}$. Высокая селективность означает, что выбирается малая часть записей, в то время как при низкой результатом является большая их часть.

Различные структуры данных ведут себя по-разному при изменении селективности запросов. Из этого следует, что выбор оптимальной структуры данных для индекса зависит от того, какова селективность запросов.

[1]ACM SIGMOD 2013 Programming Contest. `http://web.archive.org/web/20120424201336/http://wwwdb.inf.tu-dresden.de/sigmod2012contest/leaderboard/` last accessed: 26/04/2014

2. Постановка задачи

Целью данной работы является изучение влияния селективности на производительность системы многомерного индексирования при использовании различных структур данных при следующих ограничениях:

- исследуются запросы на диапазон;
- индекс должен находиться полностью в оперативной памяти;
- в качестве результата необходимо вернуть только первые 200 записей, лексикографически упорядоченных.

Для достижения этой цели были поставлены следующие задачи:

- провести обзор существующих алгоритмов для многомерного индексирования;
- реализовать тестовую систему для проведения измерений на существующем прототипе многомерного индекса;
- исследовать влияние селективности на производительность различных структур данных для многомерного индексирования при различных сценариях.

3. Обзор существующих подходов

Существует множество подходов к многомерному индексированию. Подробный обзор можно найти в [13]. Рассмотрим некоторые из возможных подходов.

- Полный просмотр данных. Такой подход хорошо работает в случае низкой селективности, поскольку в этом случае процент ошибочно просмотренных записей будет невелик. Однако при высокой селективности данный подход зачастую неэффективен.

- Сведение к одномерному случаю с помощью линейного упорядочивания с использованием кривых, заполняющих пространство (Space-Filling Curves) [21]. После этого становится возможным использование структур для индексирования одномерных данных, таких как B^+-дерево. Кривые, заполняющие пространство проходят через все целочисленные точки n-мерного пространства ровно один раз и не пересекают себя. Таким образом, все точки линейно упорядочиваются исходя из своего положения на кривой. Примеры таких кривых для двумерного пространства изображены на Рис. 1[2]:

  - (a) C-кривая — эквивалентна лексикографическому упорядочиванию;
  - (b) Z-кривая — используется в таких структурах как, UB-дерево [3] и BUB-дерево [12];
  - (c) Кривая Гильберта — используется в Гильбертовом R-дереве [17].

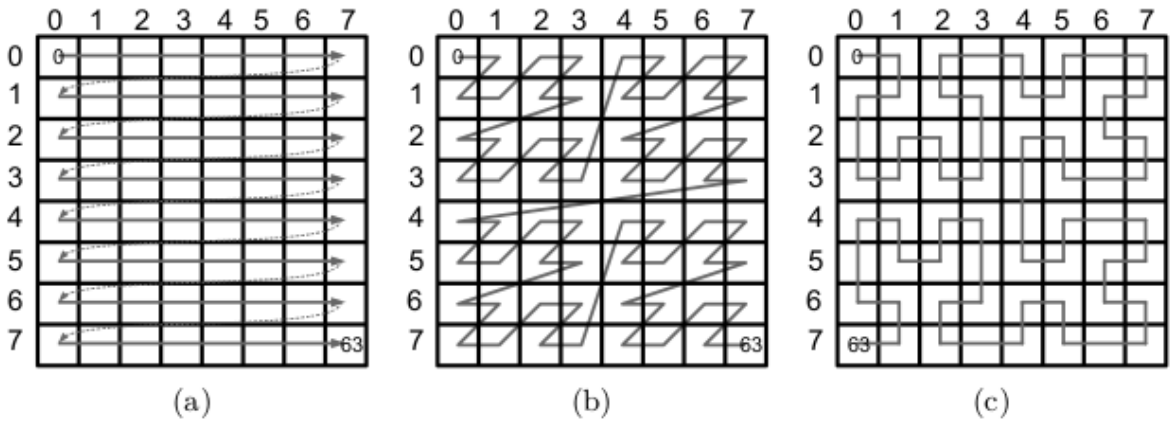

Рис. 1: Space-filling curves — примеры

[2]Изображение взято из [20]

- Подходы основанные на хэшировании. GRID-файл [22] использует деление пространства на ячейки, каждая из которых представляет из себя многомерный прямоугольник. Это позволяет осуществлять поиск записей только в необходимых ячейках, что существенно ускоряет исполнение запроса. Однако поддержка такой структуры требует дополнительных затрат. Другим примером является LSH (Locality Sensitive Hashing) [16]. Данная структура предназначена для приближенного поиска ближайших соседей за счет использования специальной хэш-функции.

- Использование древовидных структур. Поиск записей осуществляется путем обхода в глубину дерева, от корня до листьев. Примерами таких структур служат R-дерево [14], K-D-дерево [5], X-дерево [6] и т.д. K-D-дерево использует последовательное деление пространства на части по каждому измерению. На самом верхнем уровне все пространство делится на две половины по первому атрибуту, затем на следующем уровне - по второму, и так далее. В R-дереве рядом расположенные записи группируются в прямоугольники, которые в свою очередь также группируются, и т.д. Недостатком R-дерева является низкая производительность в случае пространств высокой размерности. Одной из структур, частично решающих эту проблему является X-дерево. Его особенность заключается в избежании расщепления вершин, что приводит к тому что некоторые вершины содержат больше записей, чем остальные. Также для индексирования точек из пространств высокой размерности могут использоваться техники понижения размерности, например метод главных компонент (principal component analysis, PCA).

В данной работе упор сделан на такие структуры данных, как R-дерево и B^+-дерево, в силу их широкой распространенности в сообществе разработчиков СУБД. Они имеют реализации в большинстве современных СУБД и для каждой из них существуют эффективные алгоритмы обеспечения корректности параллельного доступа.

3.1. B^+-дерево

B^+-дерево является сбалансированным деревом поиска, в котором данные хранятся только в листьях, что является его основным отличием от другой популярной

структуры данных — B-дерева [9].

Согласно [28] B^+-дерево обладает следующими свойствами.

- Дерево сбалансировано, то есть все листья находятся на одинаковой глубине.
- Внутренние вершины хранят список ключей и список указателей, причем количество последних на один больше чем первых. Каждая вершина соответствует некоторому диапазону значений, который для каждой внутренней вершины с k ключами разбивается на $k+1$ под-диапазонов, по одному на каждого ребенка. Например, предположим, что корень имеет два ключа, 100 и 200. Диапазон значений для корня в этом случае разбивается на три под-диапазона: $(-\infty, 100)$, $(100, 200)$ и $(200, \infty)$. Заметим, что ключ, хранящийся во внутренней вершине не обязательно должен присутствовать как ключ в одной из листовых вершин. Такой ключ служит лишь для определения под-диапазона.
- Листовая вершина хранит список записей, каждая запись имеет ключ и некоторое значение.
- Каждая вершина, за исключением корня, должна быть хотя бы наполовину заполнена. Например, предположим, что внутренняя вершина может хранить до p указателей на дочерние вершины, а листовая вершина - до r записей. Тогда каждая внутренняя вершина должна содержать как минимум $\lceil \frac{p}{2} \rceil$ указателей а каждая листовая вершина (за исключением корня) $\lceil \frac{r}{2} \rceil$.
- Если корень является внутренней вершиной, он должен иметь как минимум два указателя.
- Все листовые вершины организованы в двунаправленный связанный список, в порядке увеличения значений ключей.

В случае переполнения вершины при вставке новой записи, вершина расщепляется на две, причем первая половина записей остается, а вторая переносится в новую вершину. На Рис. 2[3] изображена структура B^+-дерева. Здесь корень содержит два ключа, 3 и 5, и, соответственно, три указателя на дочерние вершины.

[3]Изображение взято из `http://commons.wikimedia.org/wiki/File:Bplustree.png`

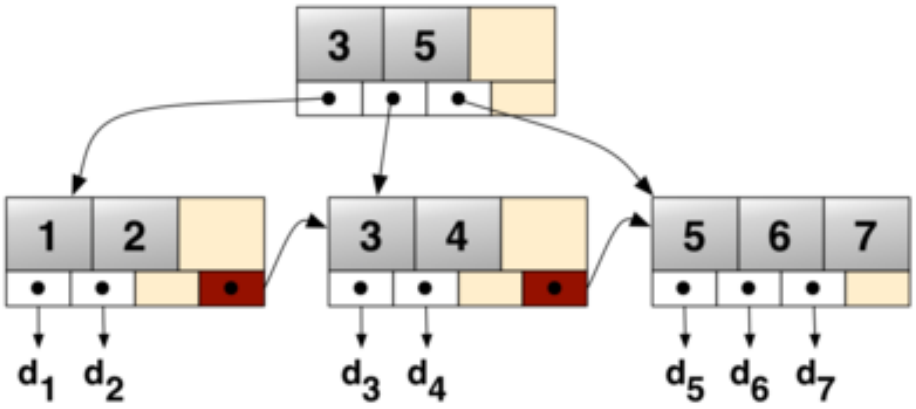

Рис. 2: B^+-дерево — пример

B^+-дерево реализовано во многих современных СУБД, таких как Oracle, Microsoft SQL Server, IBM DB2, Informix, Sybase, and MySQL и др. [28]. Оно обладает следующими преимуществами.

- Данные хранятся в упорядоченном виде. Поэтому в случае требования лексикографической упорядоченности нет необходимости в сортировке, если в качестве ключа используется конкатенация всех атрибутов.
- Существует множество алгоритмов для обеспечения корректности параллельного доступа к данному дереву, например [27], [7].

Однако B^+-дерево показывает низкую производительность в случае запросов на диапазон с высокой селективностью. Это обусловлено тем, что в таком случае неизбежен проход по нижнему уровню дерева с отбрасыванием большого количества записей, не удовлетворяющих запросу.

3.2. R-дерево

Одной из популярных структур данных для многомерного индексирования является R-дерево. Оно было предложено в 1984 г. [14], и до сих пор является предметом активных исследований. В частности, было предложено множество модификаций [4] [17] [26]. В настоящий момент существует несколько десятков вариантов R-дерева [24].

R-дерево хорошо себя зарекомендовало, и так же, как и B^+-дерево имеет реализации во многих коммерческих СУБД, например, PostgreSQL, Oracle, IBM DB2 [25].

R-дерево использует понятие минимального ограничивающего прямоугольника (minimum bounding reactangle, MBR) — минимальный прямоугольник со сторонами, параллельными осям координат, содержащий все рассматриваемые объекты.

Согласно [25] R-дерево это древовидная структура данных, определяемая парой значений (m, M) со следующими свойствами.

- Каждый лист (если он не корень) содержит от m до M записей, где $m \leq \frac{M}{2}$. Каждая запись в листе представлена в виде пары (mbr, oid), где mbr это минимальный ограничивающий прямоугольник, содержащий данный объект, а oid — идентификатор объекта.

- Количество записей хранящихся во внутренней вершине также должно принадлежать интервалу $[m, M]$. Каждая запись во внутренней вершине представляется в виде пары (mbr, p), где p — указатель на дочернюю вершину, а mbr содержит в себе mbr этого ребенка.

- Минимальное количество записей в корне — 2, за исключением случая, когда корень является листом (в этом случае он может содержать ноль или одну запись).

- Все листовые вершины находятся на одной глубине.

Пример R-дерева изображен на Рис. 3. Здесь данные хранятся в прямоугольниках E-O, а прямоугольники A-D обозначают минимальные ограничивающие прямоугольники.

R-дерево можно считать обобщением B^+-дерева:

- данные также хранятся только в листовых вершинах;

- обе структуры данных сбалансированы;

- вершины хранят ограничивающий прямоугольник, который можно считать обобщением интервала, используемого в B^+-дереве.

Однако есть и различия.

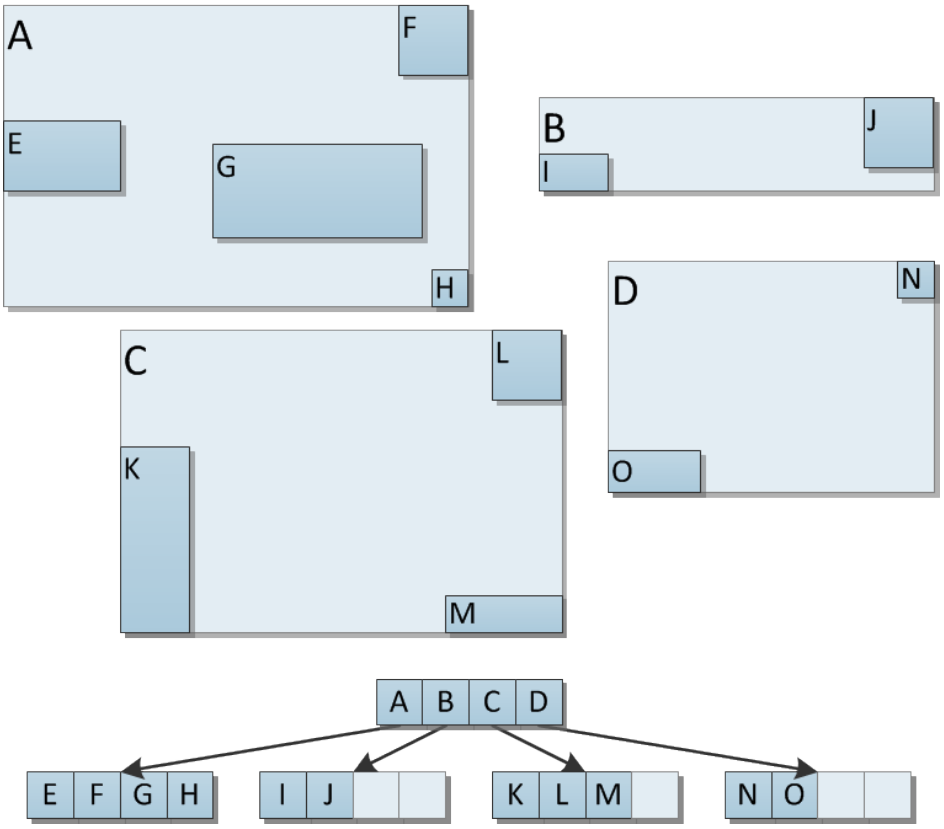

Рис. 3: R-дерево — пример

- При нахождении записей с данным ключом может потребоваться проход по нескольким веткам. Это связано с возможным пересечением ограничивающих прямоугольников.
- В случае переполнения вершины, определение оптимального способа ее расщепления является сложной задачей.
- Исходная версия, предложенная в [14] не содержит ссылок между соседними листьями.

Для R-дерева также существуют алгоритмы для обеспечения корректности параллельного доступа, например, с использованием алгоритма для структуры данных GiST (Generalized Search Tree, обобщенное дерево поиска) [18]. Последний является обобщением подобного алгоритма для B^+-дерева.

R-дерево имеет следующие преимущества:

- хорошо подходит для запросов на диапазон, однако лишь в случае высокой селективности;

- имеются алгоритмы для обеспечения корректности параллельного доступа.

Из недостатков следует отметить следующие:

- низкая производительность на данных высокой размерности, так как R-дерево не предназначено для таких случаев [24];
- низкая производительность в случае низкой селективности запросов на диапазон.

4. Тестовая система для проведения измерений

Для исследования влияния селективности на производительность запросов к многомерным структурам данных можно предложить несколько подходов. Первый из них аналитический, то есть построение модели и дальнейшая оценка производительности. Однако для получения реалистичной модели, требуется учет множества параметров, в частности параллельного исполнения запросов. В связи с этим, такой подход очень трудоемок, и по этой причине в данной работе был выбран другой вариант, а именно экспериментальное сравнение, для которого необходимо наличие прототипа многомерного индекса с поддержкой транзакций.

Среди существующих прототипов следует отметить следующие.

- Elki [1] — фреймворк, предназначенный в основном для исследования алгоритмов кластеризации. Ведется активная разработка, однако отсутствует поддержка корректности параллельного доступа.
- Amdb [19] — содержит множество утилит (визуализатор для отображения структуры индекса, сборщик статистики, поддерживающий большое количество различных метрик и др.), однако также отсутствует поддержка корректности параллельного доступа. На текущий момент разработка неактивна.
- Прототип, описанный в [2] — предназначен для изучения эффективности I/O и, как следствие, не подходит для случая индекса в оперативной памяти. В настоящий момент разработка неактивна.

В силу указанных выше недостатков имеющихся систем был выбран существующий прототип системы многомерного индексирования с поддержкой транзакций, который разрабатывается на математико-механическом факультете СПбГУ[4] и содержит необходимые для исследования реализации B^+-дерева и R-дерева.

4.1. Описание прототипа многомерного индекса

Выбранный прототип многомерного индекса обладает следующими свойствами.

- Реализация B^+-дерева. Для обеспечения корректности параллельного доступа используется алгоритм, описанный в [27].

[4]работа частично поддержана грантом РФФИ №12-07-31050

- Реализация R-дерева с помощью структуры данных GiST [15] - популярного шаблона для реализации древовидных структур данных (например, такой подход используется в PostgreSQL). Обеспечение корректности параллельного доступа реализовано с помощью представленного в статье [18] алгоритма.
- Уровень изоляции транзакций — чтение фиксированных данных (read committed).
- Индекс находится в оперативной памяти.

Для поддержки выборки первых k лексикографически упорядоченных записей, удовлетворяющих запросу, в прототипе используется следующая модификация R-дерева: все записи в листовых вершинах хранятся в упорядоченной виде. При поиске сначала находятся все листовые вершины, в которых содержатся записи, удовлетворяющие запросу, затем используется сортировка слиянием для получения упорядоченной выдачи. Преимущество такого подхода состоит в том, что для выдачи первых k записей нет необходимости полной сортировки результата.

4.2. Используемые методы и технологии

Тестовая система реализована на языке C/C++ и работает под управлением ОС linux. При реализации были использованы следующие методы и технологии:

- существующий прототип многомерного индекса, описанный выше;
- библиотека pthreads для работы с потоками;
- valgrind, для тестирования и отладки, в частности, такие утилиты как drd, memcheck, helgrind;
- subversion в качестве системы контроля версий.

4.3. Архитектура

В данной работе тестирование производилось на различных синтетических данных. Сначала индекс заполнялся некоторыми начальными данными, затем исполнялись запросы.

Тестовая система состоит из двух компонент:

- модуль для генерации начальных данных и запросов;
- модуль для проведения измерений.

4.3.1. Генератор запросов и начальных данных

Данная компонента служит для генерации начальных записей и запросов к индексу. На вход ей подается xml-файл с описанием нагрузки, на выходе два файла: первый описывает начальные записи, второй — запросы. Пример входного файла изображен на Рис. 4. На текущий момент поддерживается задание следующих параметров:

```
<format version="1.0"/>
<workload_description>
  <queries_properties transactions_amount="1000">
    <range portion="100" selectivity="0.0001"/>
    <point portion="0"/>
    <update portion="0"/>
    <insert portion="0"/>
    <delete portion="0"/>
  </queries_properties>
  <index_properties name="test" size="32 MB" payload_size="16">
    <attributes dimensions="2" data_distribution="COMBINED">
      <attribute distribution="UNIFORM" min="1" max="100000"/>
      <attribute distribution="UNIFORM" min="1" max="100000"/>
    </attributes>
  </index_properties>
</workload_description>
```

Рис. 4: Пример описания нагрузочных данных

- размер индекса, то есть суммарный размер всех начальных записей;
- распределение данных для каждого атрибута;
- количество запросов;
- селективность запросов на диапазон.

Можно предложить несколько подходов к генерации запросов на диапазон с заданной селективностью. Первый — зная распределение данных, для каждого прямоугольника можно с некоторой вероятностью сказать, какой процент точек в него

попадает. Таким образом, можно сразу генерировать прямоугольники с заданной селективностью. Однако в данном случае для каждого распределения данных необходима реализация своего алгоритма.

В данной работе используется другой подход. Его суть заключается в последовательном приближении к заданной селективности. Сначала генерируется случайный прямоугольник, затем, если значение селективности при соответствующем запросе меньше, чем заданное, то генерируется прямоугольник, включающих исходный, если же больше, то содержащийся в нем. Данный процесс повторяется до тех пор, пока не будет получен запрос с заданной селективностью. Преимущество данного алгоритма в том, что он работает для большинства распределений данных.

4.3.2. Модуль для проведения измерений

Данная компонента служит для непосредственного измерения производительности на фиксированном наборе начальных данных и запросов:

- считывается файл с начальными данными и происходит заполнение индекса;
- warm-up: запросы исполняются в течение k секунд без подсчета статистики;
- исполнение запросов со сбором статистики в течении n секунд.

Поддерживается задание параметров k и n, а также количества потоков, одновременно обращающихся к индексу. Собирается же следующая статистика:

- время заполнения индекса;
- количество выполненных запросов (суммарное по всем потокам, и по каждому потоку отдельно);
- количество транзакций в секунду (суммарное по всем потокам, и по каждому потоку отдельно).

5. Исследование влияния селективности

5.1. Описание тестового стенда

Для проведения экспериментов использовался прототип многомерного индекса и реализованная тестовая система. В качестве тестового стенда были использованы следующие конфигурации:

- Конфигурация 1:
  - аппаратное обеспечение: Intel Core i7-2630QM, 2.00 GHz, 6GB RAM;
  - программное обеспечение: x86_64 GNU/Linux, ядро 3.11.0-12-generic.
- Конфигурация 2:
  - аппаратное обеспечение: Intel Core i7-3770, 3.40GHz, 8GB RAM;
  - программное обеспечение: x86_64 GNU/Linux, ядро 3.8.0-26-generic.

Эксперименты с использованием более производительной конфигурации можно найти в статье [23]. Однако на момент написания данной работы она была недоступна, поэтому эксперименты проводились только на вышеупомянутых конфигурациях.

5.2. Проведение экспериментов

Для исследования влияния селективности на производительность была проведена серия экспериментов. В них исследовалась пропускная способность прототипа при использовании одной из двух структур данных (B^+-дерево и R-дерево) при различных значениях селективности запросов. Были зафиксированы следующие параметры:

- для B^+-дерева максимальное количество ссылок в вершине (fan-out) равно 512, для R-дерева 32;
- для расщепления вершин в R-дереве используется квадратичный алгоритм, описанный в оригинальной статье [14];
- в качестве запросов использовались запросы на диапазон;

- в качестве результата запроса возвращались только первые 200 лексикографически упороченных записей.

При проведении экспериментов варьировались следующие параметры.

- Размерность: 2, 4, 6 и 8. Пространства большей размерности не исследуются в силу того, что R-дерево показывают низкую производительность в таких случаях [24].
- Размер индекса: 32MB.
- Распределение данных: равномерное, нормальное, закон Ципфа.
- Селективность: от 0.001% до 0.01%. Более высокие значения селективности не рассматриваются из-за того, что при них становится выгоднее использовать полный просмотр. В данной работе рассматриваются запросы, в которых диапазон выборки по каждому атрибуту примерно одинаков. Вырожденные случаи, когда, например, по первому измерению выбирается все, а по всем остальным почти ничего, не рассматриваются.

5.3. Результаты измерений

Графики, описывающие результаты экспериментов с использованием первой конфигурации тестового стенда, представлены в приложении 1. Для иллюстрации использованы двойные логарифмические оси. По оси абсцисс отложены значения селективности, по оси ординат — соответствующие значения пропускной способности. Для удобства графики для B^+-дерева и R-дерева при одинаковом наборе параметров представлены на одном рисунке. Значения пропускной способности оценивались с 95% доверительным интервалом. Можно заметить следующее.

- При увеличении значения селективности производительность R-дерева падает. Это связано с тем, что требуется сортировать больше записей, чтобы найти первые k.
- Производительность B^+-дерева падает при уменьшении значения селективности. Это связано с тем, что требуется просматривать больше записей, не удовлетворяющих запросу.

- Существует порог значения селективности, при котором B^+-дерево и R-дерево показывают одинаковый результат. Графики наглядно это демонстрируют. При значении селективности выше этого порога, целесообразно выбирать B^+-дерево для достижения лучшей пропускной способности, при значении ниже — R-дерево.

- В двойных логарифмических осях график зависимости пропускной способности от значения селективности выглядит как прямая линия. Это наблюдается при всех наборах параметров, как для B^+-дерева, так и для R-дерева. Это приводит к гипотезе о том, что пропускная способность может быть выражена с помощью степенного закона: $P = a * S^b$, где P обозначает пропускную способность, S - селективность запросов, a и b - параметры, зависящие от используемого тестового стенда, а также размерности, распределения данных и пр. Данная гипотеза была принята на уровне значимости 1%, значения параметра b представлены в приложении 2.

- Для проверки воспроизводимости полученных результатов, все эксперименты были повторно проведены с использованием второй конфигурации тестового стенда. Значения параметра b, показывающие сопоставимость результатов при использовании обеих конфигураций, представлены в приложении 2.

- Увеличение размерности приводит к уменьшению параметра b в случае R-дерева, то есть чем больше размерность, тем меньше влияние селективности. В случае B^+-дерева наоборот: с увеличением размерности влияние селективности растет. Указанное поведение проявляется на всех исследуемых распределениях данных.

6. Заключение

В ходе данной дипломной работы было проведено исследование влияния селективности на производительность различных многомерных индексов:

- проведен обзор существующих алгоритмов для многомерного индексирования;
- реализована тестовая система для проведения измерений на существующем прототипе многомерного индекса с поддержкой задания различных параметров;
- исследовано влияние селективности на производительность таких структур данных для многомерного индексирования, как B^+-дерево, R-дерево.

Также по теме работы была опубликована статья на конференции SYRCoDIS'13: "To Sort or not to Sort: The Evaluation of R-Tree and B^+-tree in Transactional Environment with Ordered Result Set Requirement." [23].

Работа выполнялась в рамках гранта РФФИ №12-07-31050.

Список литературы

[1] Achtert Elke, Kriegel Hans-Peter, Zimek Arthur. ELKI: A Software System for Evaluation of Subspace Clustering Algorithms // Scientific and Statistical Database Management / Ed. by Bertram Ludäscher, Nikos Mamoulis. — Springer Berlin Heidelberg, 2008. — Vol. 5069 of Lecture Notes in Computer Science. — P. 580–585. — URL: http://dx.doi.org/10.1007/978-3-540-69497-7_41.

[2] Arge Lars, Procopiuc Octavian, Scott Vitter Jeffrey. Implementing I/O-efficient Data Structures Using TPIE // Algorithms — ESA 2002 / Ed. by Rolf Möhring, Rajeev Raman. — Springer Berlin Heidelberg, 2002. — Vol. 2461 of Lecture Notes in Computer Science. — P. 88–100. — URL: http://dx.doi.org/10.1007/3-540-45749-6_12.

[3] Bayer Rudolf. The Universal B-Tree for Multidimensional Indexing: General Concepts // Proceedings of the International Conference on Worldwide Computing and Its Applications. — WWCA '97. — London, UK, UK : Springer-Verlag, 1997. — P. 198–209. — URL: http://dl.acm.org/citation.cfm?id=645965.674403.

[4] Beckmann Norbert, Seeger Bernhard. A Revised R*-tree in Comparison with Related Index Structures // Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. — SIGMOD '09. — New York, NY, USA : ACM, 2009. — P. 799–812. — URL: http://doi.acm.org/10.1145/1559845.1559929.

[5] Bentley Jon Louis. Multidimensional Binary Search Trees Used for Associative Searching // Commun. ACM. — 1975. — sep. — Vol. 18, no. 9. — P. 509–517. — URL: http://doi.acm.org/10.1145/361002.361007.

[6] Berchtold Stefan, Keim Daniel A., Kriegel Hans-Peter. The X-tree: An Index Structure for High-Dimensional Data // Proceedings of the 22th International Conference on Very Large Data Bases. — VLDB '96. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1996. — P. 28–39. — URL: http://dl.acm.org/citation.cfm?id=645922.673502.

[7] Cache-Conscious Concurrency Control of Main-Memory Indexes on Shared-Memory Multiprocessor Systems / Sang K. Cha, Sangyong Hwang, Kihong Kim, Keunjoo Kwon // Proceedings of the 27th International Conference on Very Large Data Bases. — VLDB '01. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001. — P. 181–190. — URL: http://dl.acm.org/citation.cfm?id=645927.672375.

[8] Chaudhuri Surajit, Narasayya Vivek. Self-tuning Database Systems: A Decade of Progress // Proceedings of the 33rd International Conference on Very Large Data Bases. — VLDB '07. — VLDB Endowment, 2007. — P. 3–14. — URL: http://dl.acm.org/citation.cfm?id=1325851.1325856.

[9] Comer Douglas. Ubiquitous B-Tree // ACM Comput. Surv. — 1979. — jun. — Vol. 11, no. 2. — P. 121–137. — URL: http://doi.acm.org/10.1145/356770.356776.

[10] Efficient Top-k Queries for Orthogonal Ranges / Saladi Rahul, Prosenjit Gupta, Ravi Janardan, K. S. Rajan // Proceedings of the 5th International Conference on WALCOM: Algorithms and Computation. — WALCOM'11. — Berlin, Heidelberg : Springer-Verlag, 2011. — P. 110–121. — URL: http://dl.acm.org/citation.cfm?id=1966169.1966187.

[11] The End of an Architectural Era: (It's Time for a Complete Rewrite) / Michael Stonebraker, Samuel Madden, Daniel J. Abadi et al. // Proceedings of the 33rd International Conference on Very Large Data Bases. — VLDB '07. — VLDB Endowment, 2007. — P. 1150–1160. — URL: http://dl.acm.org/citation.cfm?id=1325851.1325981.

[12] Fenk Robert. The BUB-Tree // IN VLDB '02, PROCEEDINGS OF 28TH INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, HONG KONG. — Morgan Kaufman Publishers, 2002.

[13] Gaede Volker, Günther Oliver. Multidimensional Access Methods // ACM Comput. Surv. — 1998. — jun. — Vol. 30, no. 2. — P. 170–231. — URL: http://doi.acm.org/10.1145/280277.280279.

[14] Guttman Antonin. R-trees: A Dynamic Index Structure for Spatial Searching // Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data. — SIGMOD '84. — New York, NY, USA : ACM, 1984. — P. 47–57. — URL: http://doi.acm.org/10.1145/602259.602266.

[15] Hellerstein Joseph M., Naughton Jeffrey F., Pfeffer Avi. Generalized Search Trees for Database Systems // Proceedings of the 21th International Conference on Very Large Data Bases. — VLDB '95. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1995. — P. 562–573. — URL: http://dl.acm.org/citation.cfm?id=645921.673145.

[16] Indyk Piotr, Motwani Rajeev. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality // Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. — STOC '98. — New York, NY, USA : ACM, 1998. — P. 604–613. — URL: http://doi.acm.org/10.1145/276698.276876.

[17] Kamel Ibrahim, Faloutsos Christos. Hilbert R-tree: An Improved R-tree Using Fractals // Proceedings of the 20th International Conference on Very Large Data Bases. — VLDB '94. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1994. — P. 500–509. — URL: http://dl.acm.org/citation.cfm?id=645920.673001.

[18] Kornacker Marcel, Mohan C., Hellerstein Joseph M. Concurrency and Recovery in Generalized Search Trees // SIGMOD Rec. — 1997. — jun. — Vol. 26, no. 2. — P. 62–72. — URL: http://doi.acm.org/10.1145/253262.253272.

[19] Kornacker Marcel, Shah Mehul, Hellerstein Joseph M. AMDB: An Access Method Debugging Tool // SIGMOD Rec. — 1998. — jun. — Vol. 27, no. 2. — P. 570–571. — URL: http://doi.acm.org/10.1145/276305.276384.

[20] Krizka Filip, Krátký Michal, Baca Radim. On Support of Ordering in Multidimensional Data Structures. // ADBIS (Local Proceedings) / Ed. by Mirjana Ivanovic, Bernhard Thalheim, Barbara Catania, Zoran Budimac. — Vol. 639 of CEUR Workshop Proceedings. — CEUR-WS.org, 2010. — P. 165–179.

[21] Lawder Jonathan K., King Peter J. H. Using Space-Filling Curves for Multi-dimensional Indexing // Proceedings of the 17th British National Conferenc on Databases: Advances in Databases. — BNCOD 17. — London, UK, UK : Springer-Verlag, 2000. — P. 20–35. — URL: http://dl.acm.org/citation.cfm?id=646102.681186.

[22] Nievergelt J., Hinterberger Hans, Sevcik Kenneth C. The Grid File: An Adaptable, Symmetric Multikey File Structure // ACM Trans. Database Syst. — 1984. — mar. — Vol. 9, no. 1. — P. 38–71. — URL: http://doi.acm.org/10.1145/348.318586.

[23] Proceedings of the Ninth Spring Researchers Colloquium on Databases and Information Systems, Kazan, Russia, May 31, 2013 / Ed. by Natalia Vassilieva, Denis Turdakov, Vladimir Ivanov. — Vol. 1031 of CEUR Workshop Proceedings, CEUR-WS.org, 2013.

[24] R-Tree (and Family) / Apostolos N. Papadopoulos, Antonio Corral, Alexandros Nanopoulos, Yannis Theodoridis // Encyclopedia of Database Systems / Ed. by LING LIU, M.TAMER ÖZSU. — Springer US, 2009. — P. 2453–2459. — URL: http://dx.doi.org/10.1007/978-0-387-39940-9_300.

[25] R-Trees: Theory and Applications (Advanced Information and Knowledge Processing) / Yannis Manolopoulos, Alexandros Nanopoulos, Apostolos N. Papadopoulos, Y. Theodoridis. — Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2005. — ISBN: 1852339772.

[26] Sellis Timos K., Roussopoulos Nick, Faloutsos Christos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects // Proceedings of the 13th International Conference on Very Large Data Bases. — VLDB '87. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1987. — P. 507–518. — URL: http://dl.acm.org/citation.cfm?id=645914.671636.

[27] Weikum Gerhard, Vossen Gottfried. Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001. — ISBN: 1-55860-508-8.

[28] Zhang Donghui, Baclawski KennethPaul, J. Tsotras Vassilis. B+-Tree // Encyclopedia of Database Systems / Ed. by LING LIU, M.TAMER ÖZSU. — Springer US, 2009. — P. 197–200. — URL: http://dx.doi.org/10.1007/978-0-387-39940-9_739.

7. Приложение 1

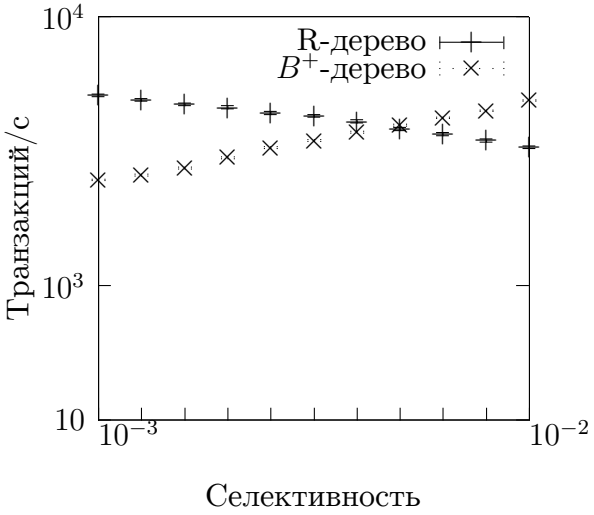

Рис. 5: Равномерное распределение, размерность 2

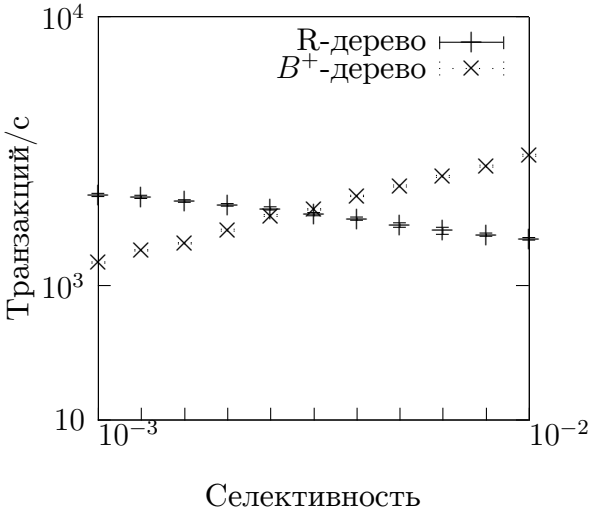

Рис. 6: Равномерное распределение, размерность 4

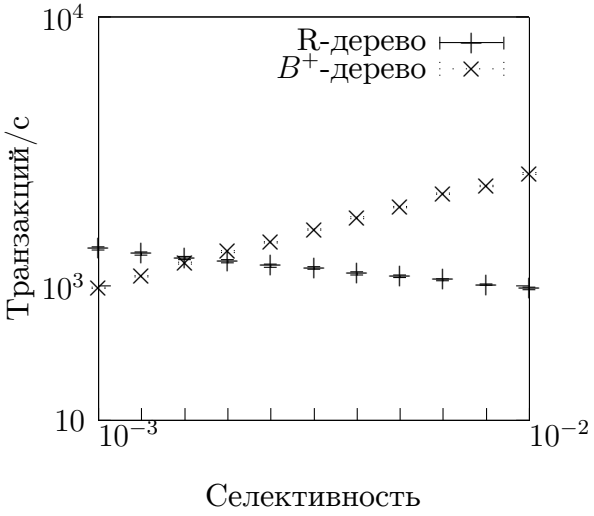

Рис. 7: Равномерное распределение, размерность 6

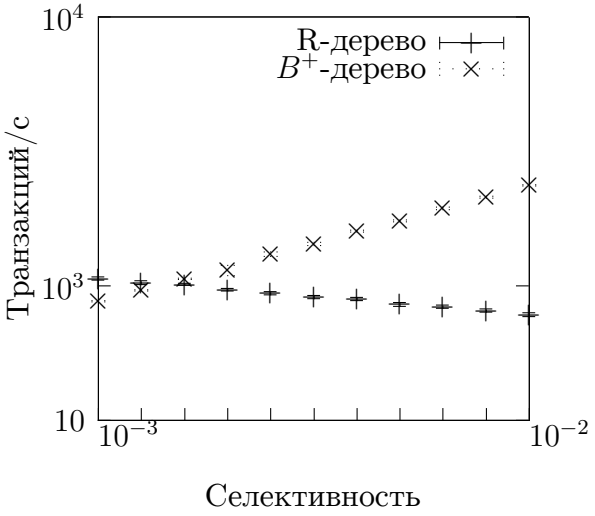

Рис. 8: Равномерное распределение, размерность 8

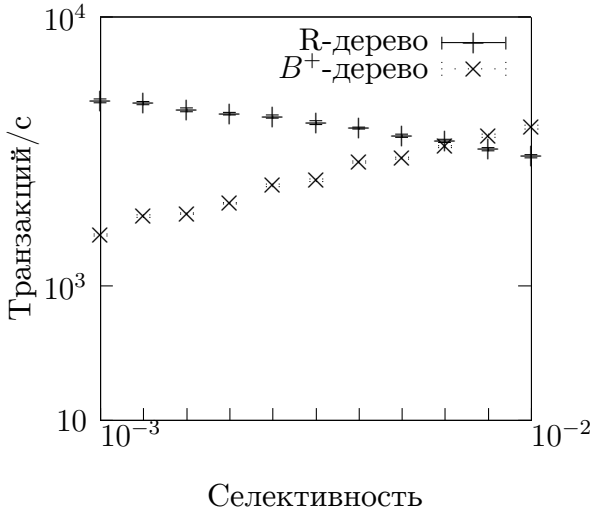

Рис. 9: Нормальное распределение, размерность 2

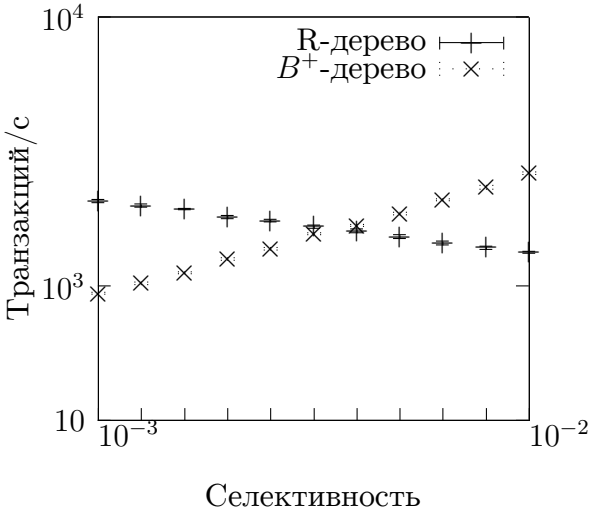

Рис. 10: Нормальное распределение, размерность 4

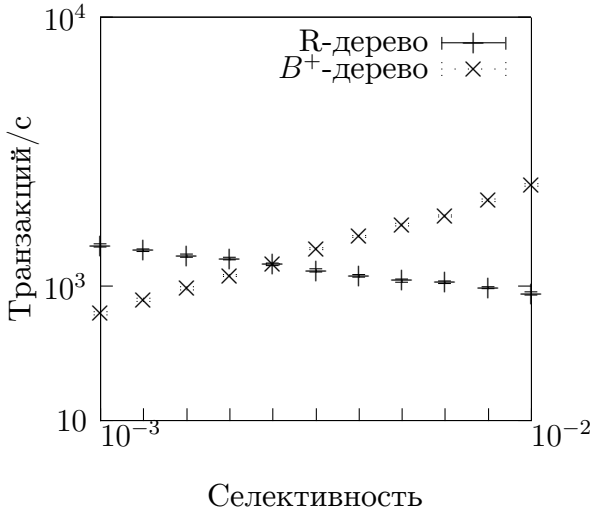

Рис. 11: Нормальное распределение, размерность 6

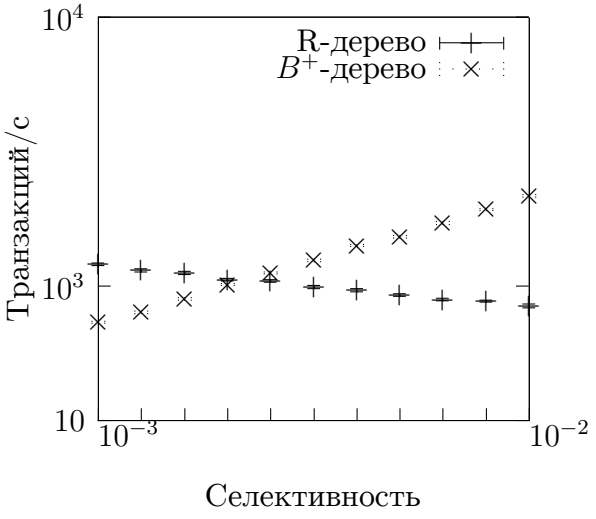

Рис. 12: Нормальное распределение, размерность 8

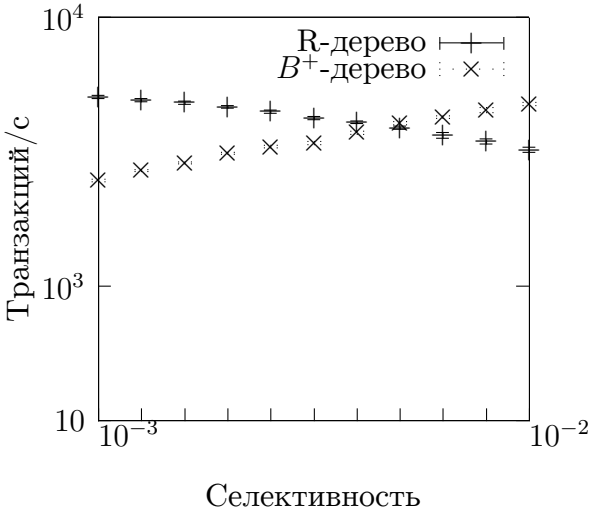

Рис. 13: Закон Ципфа, размерность 2

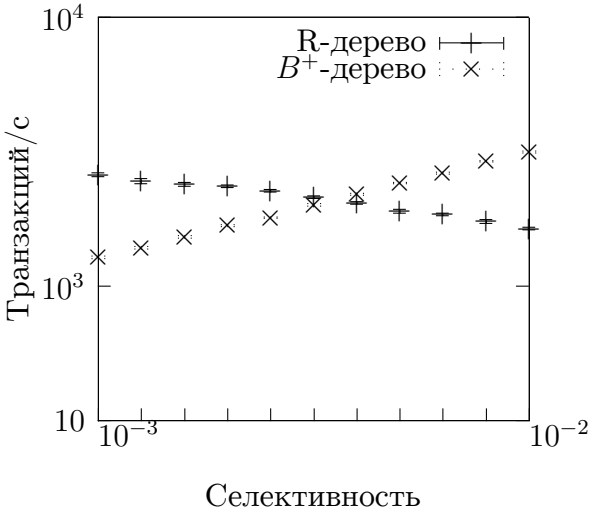

Рис. 14: Закон Ципфа, размерность 4

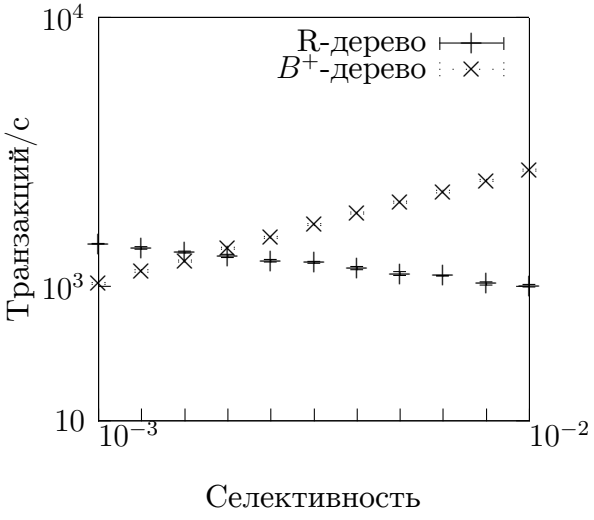

Рис. 15: Закон Ципфа, размерность 6

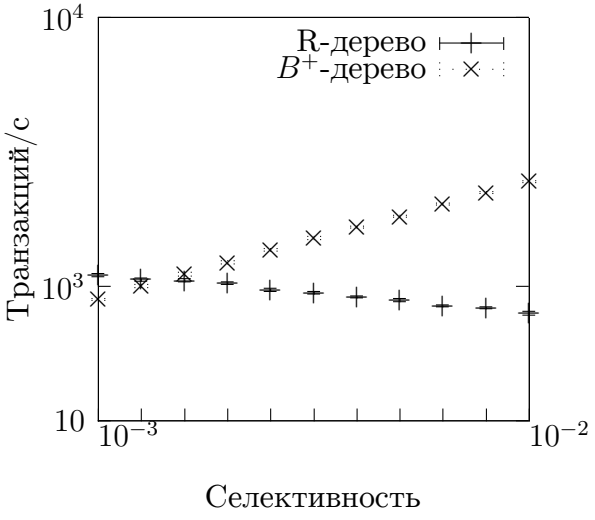

Рис. 16: Закон Ципфа, размерность 8

8. Приложение 2

Распределение	2	4	6	8
Равномерное	0.60 ± 0.03	0.80 ± 0.03	0.85 ± 0.02	0.87 ± 0.02
Нормальное	0.80 ± 0.06	0.89 ± 0.03	0.94 ± 0.03	0.94 ± 0.02
Закон Ципфа	0.57 ± 0.03	0.79 ± 0.02	0.85 ± 0.02	0.87 ± 0.02

Таблица 1: Значения параметра b для B^+-дерева на 1-ой конфигурации

Распределение	2	4	6	8
Равномерное	0.62 ± 0.02	0.79 ± 0.03	0.86 ± 0.03	0.88 ± 0.01
Нормальное	0.81 ± 0.06	0.92 ± 0.04	0.99 ± 0.04	1.00 ± 0.03
Закон Ципфа	0.57 ± 0.04	0.79 ± 0.02	0.86 ± 0.02	0.86 ± 0.03

Таблица 2: Значения параметра b для B^+-дерева на 2-ой конфигурации

Распределение	2	4	6	8
Равномерное	-0.41 ± 0.05	-0.34 ± 0.02	-0.31 ± 0.02	-0.30 ± 0.01
Нормальное	-0.42 ± 0.05	-0.41 ± 0.02	-0.37 ± 0.03	-0.32 ± 0.03
Закон Ципфа	-0.40 ± 0.04	-0.38 ± 0.02	-0.34 ± 0.02	-0.31 ± 0.02

Таблица 3: Значения параметра b для R-дерева на 1-ой конфигурации

Распределение	2	4	6	8
Равномерное	-0.40 ± 0.04	-0.35 ± 0.01	-0.31 ± 0.02	-0.28 ± 0.01
Нормальное	-0.43 ± 0.05	-0.39 ± 0.02	-0.37 ± 0.02	-0.33 ± 0.03
Закон Ципфа	-0.40 ± 0.04	-0.40 ± 0.02	-0.34 ± 0.02	-0.31 ± 0.02

Таблица 4: Значения параметра b для R-дерева на 2-ой конфигурации