НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Факультет компьютерных наук
Департамент программной инженерии

КОМПОНЕНТ-РАСШИРЕНИЕ РСУБД SQLITE ДЛЯ ИНДЕКСИРОВАНИЯ ДАННЫХ МОДИФИКАЦИЯМИ B-ДЕРЕВЬЕВ

Выпускная квалификационная работа

Исполнитель: Ригин А.М., студент группы БПИ153

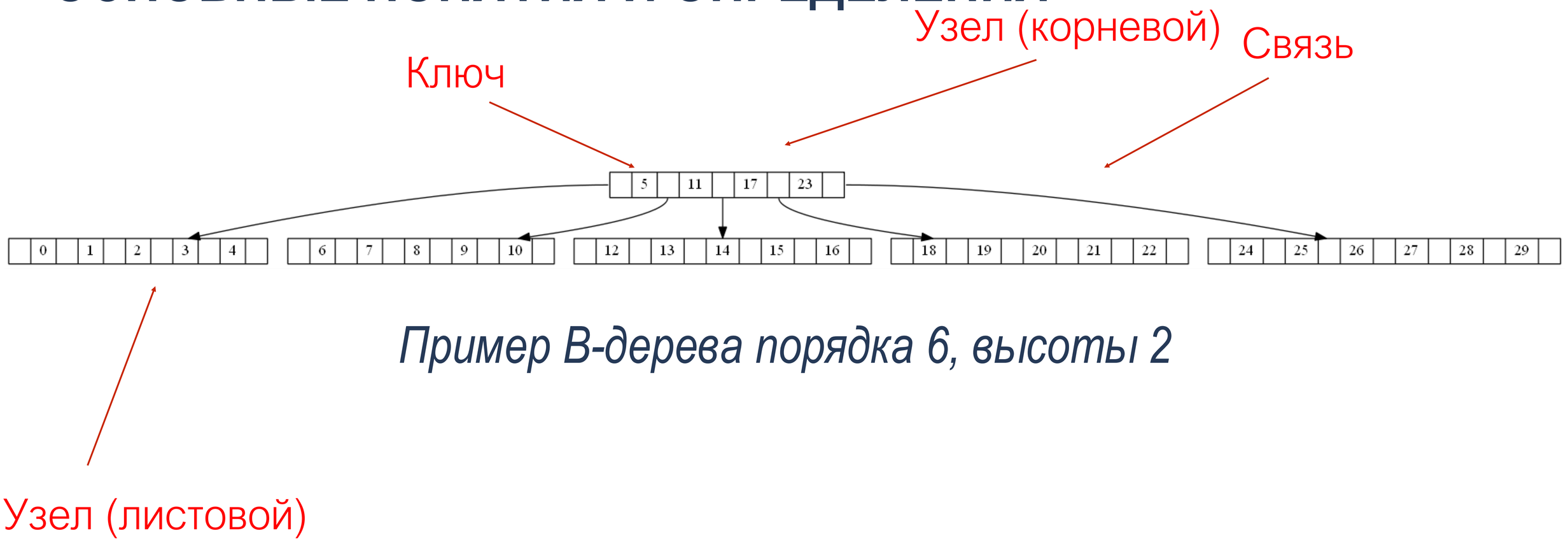
Научный руководитель: Шершаков С.А.,
ст. преп. ДПИ ФКН, н.с. НУЛ ПОИС ФКН

Москва, 2019

ПРЕДМЕТНАЯ ОБЛАСТЬ

- Сильно ветвящиеся деревья: B-деревья, B^+-деревья, B^*-деревья и B^{*+}-деревья
- Эмпирическая оценка сложности основных операций и объёма занимаемой памяти
- Индексы в СУБД
- SQLite – встраиваемая реляционная СУБД
- Использование сильно ветвящихся деревьев в качестве индекса в SQLite

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ



Пример B-дерева порядка 6, высоты 2

[5] Ригин А.М. Исследование эффективности сильно ветвящихся деревьев в задаче индексирования структурированных данных

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

- **Сильно ветвящееся дерево** (структура данных) – такое дерево, которое может содержать в одном узле более одного элемента с ключом и более одной связи с дочерними узлами.
- **B-дерево** – Сильно ветвящееся дерево. B-дерево построено так, что если некоторый нелистовой узел содержит k ключей, то у данного узла $k+1$ потомков, и для любого i, такого, что $1 \leq i \leq k+1$, верно, что все ключи в i-м потомке данного узла не меньше, чем i-й ключ данного узла, и не больше, чем $i+1$-й ключ данного узла [4].
- **Порядок B-дерева** – такое число t, что для любого некорневого узла дерева верно неравенство: $t-1 \leq k \leq 2t-1$, где k – число ключей в узле. Корневой узел для непустого B-дерева содержит $1 \leq k \leq 2t-1$ ключей, для пустого B-дерева – 0 ключей [4].
- B-дерево является **сбалансированным деревом**, поэтому его высота будет равна $O(log_t\, n)$, где n – число ключей в дереве [4].

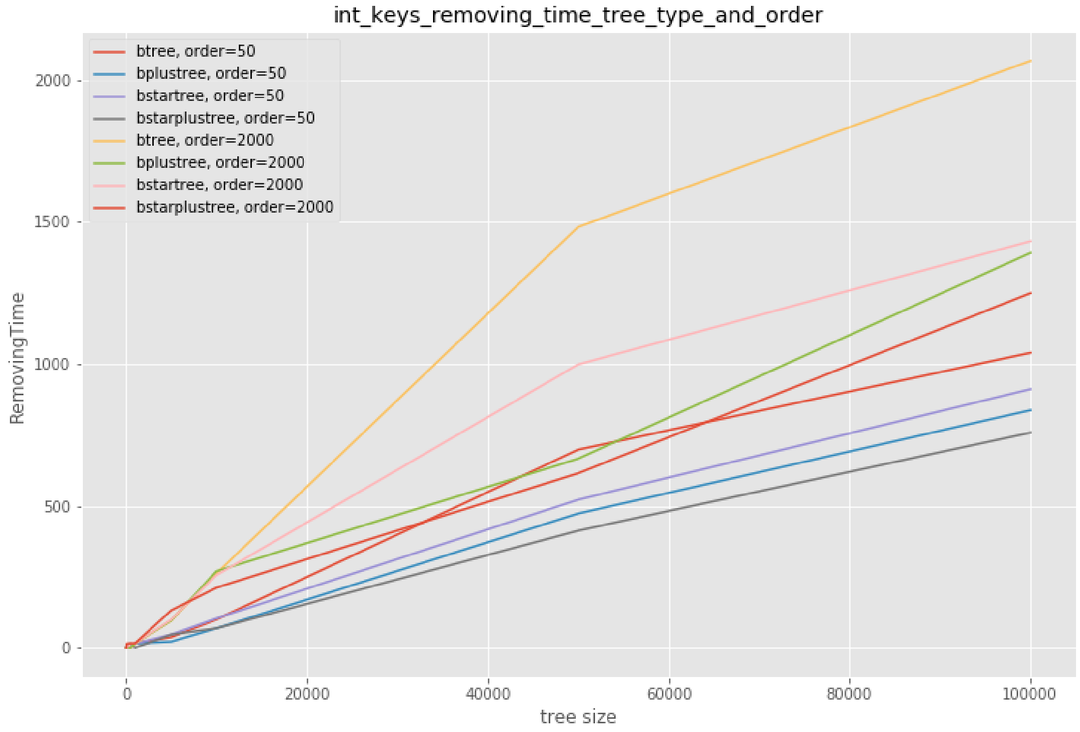
[4] Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. Алгоритмы: построение и анализ. 3-е изд.

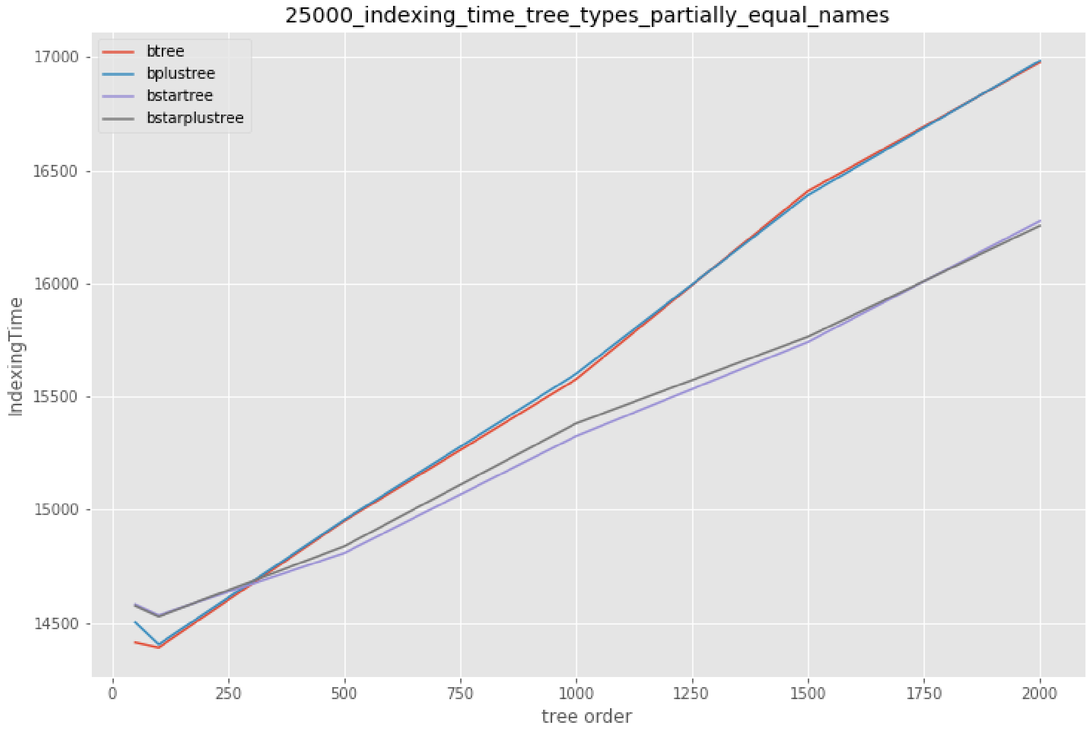
ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

- **B⁺-дерево** – модификация B-дерева. В B⁺-дереве настоящие ключи хранятся лишь в листьях дерева, а во внутренних узлах хранятся лишь ключи-маршрутизаторы, необходимые для поиска по дереву. Листья в B⁺-дереве содержат $t \leq n \leq 2t$ ключей, где t – порядок дерева, ограничения для внутренних узлов такие же, как и в B-дереве. [1] [2].
- **B*-дерево** – модификация B-дерева. Каждый узел заполняется не менее, чем на 2/3, а не 1/2. [1].
- **B*⁺-дерево** – модификация B-дерева, **разработанная в рамках выполнения курсовой работы за 3 курс [5]**. Представляет собой совмещение B⁺-дерева и B*-дерева: вершины заполняются не менее, чем на 2/3, как в B*-дереве, при этом, как в B⁺-дереве, реальные данные хранятся только в листьях, в остальных вершинах находятся лишь ключи-маршрутизаторы.

[1] Comer D. The Ubiquitous B-Tree
[2] Kerttu Pollari-Malmi. B+-trees: https://www.cs.helsinki.fi/u/mluukkai/tirak2010/B-tree.pdf
[5] Ригин А.М. Исследование эффективности сильно ветвящихся деревьев в задаче индексирования структурированных данных

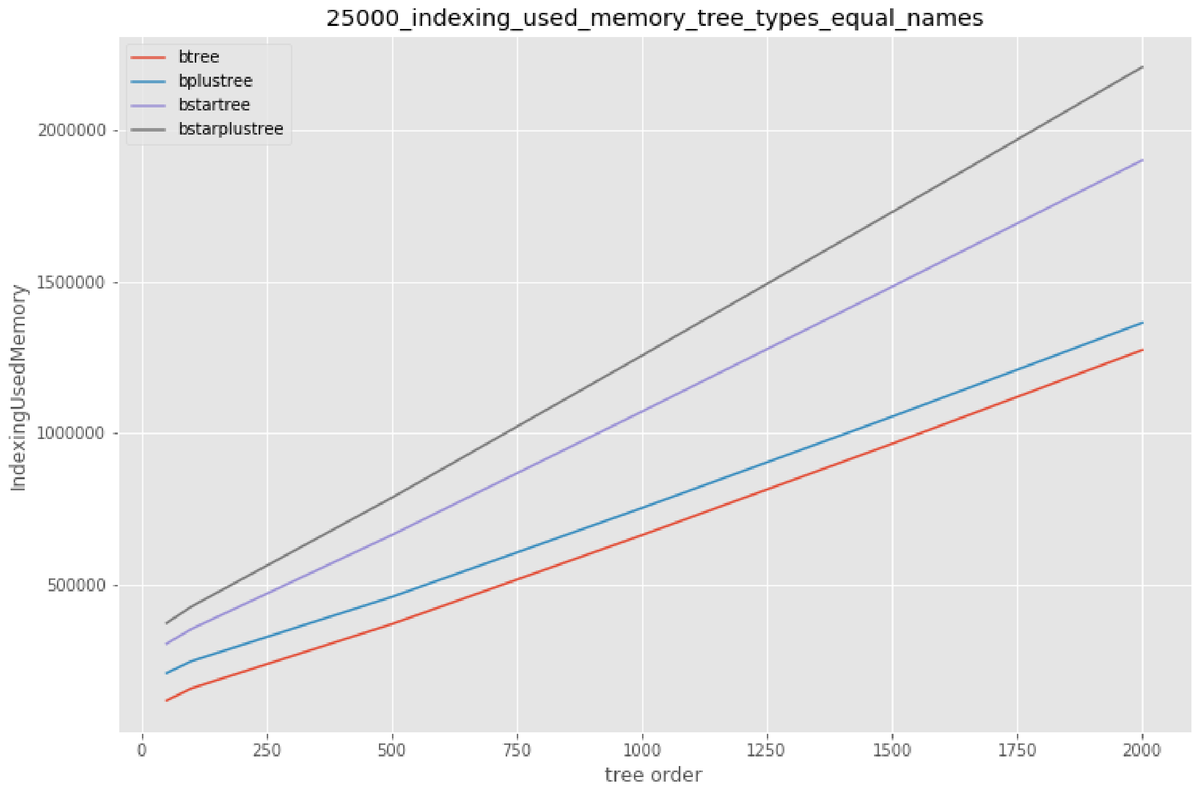
ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ





[5] Ригин А.М. Исследование эффективности сильно ветвящихся деревьев в задаче индексирования структурированных данных

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ



[5] Ригин А.М. Исследование эффективности сильно ветвящихся деревьев в задаче индексирования структурированных данных

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

- **СУБД** – система управления базами данных
- **РСУБД** – реляционная СУБД
- **SQLite** – РСУБД с открытым исходным кодом (написана на языке C)
- **Расширение SQLite** – библиотека динамического подключения, расширяющая функционал РСУБД SQLite и предоставляющая новые функции [3]

[3] https://www.sqlite.org/loadext.html

АКТУАЛЬНОСТЬ ТЕМЫ И СУЩЕСТВУЮЩИЕ РЕШЕНИЯ

- **Актуальность темы**
  - ✓ В настоящее время растут объёмы обрабатываемых данных
  - ✓ Необходимо разрабатывать новые эффективные подходы к индексации данных в СУБД
  - ✓ СУБД SQLite содержит небольшое число способов индексирования данных – актуально добавление новых
- **Существующие решения**
  - ✓ B-дерево является способом индексации по умолчанию в SQLite
  - ✓ Существует ряд расширений для SQLite, добавляющих, например, индексирование при помощи R-дерева
  - ✓ Расширений с B^+-деревом, B^*-деревом и B^{*+}-деревом не обнаружено

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

- **Цель работы**
  - ✓ Разработать расширение для SQLite, позволяющее использовать в качестве индекса в данной СУБД модификации B-дерева: B^+-дерево, B^*-дерево и B^{*+}-дерево
- **Задачи работы**
  - ✓ Реализовать API на C для имеющейся C++-библиотеки алгоритмов над сильно ветвящимися деревьями
  - ✓ Разработать интерфейс подключения данной библиотеки в качестве расширения к SQLite, реализовать собственно расширение для SQLite
  - ✓ Реализовать в расширении функциональность для вывода графического представления используемого дерева и основной информации о нём
  - ✓ Разработать алгоритм выбора структуры данных в качестве индекса
  - ✓ Разработать техническую документацию в соответствии с ЕСПД

АЛГОРИТМ ВЫБОРА ИНДЕКСИРУЮЩЕЙ СТРУКТУРЫ ДАННЫХ

- Выбирает из модификаций B-дерева (B^+-дерева, B^*-дерева и B^{*+}-дерева)
- Запускается при каждой операции с таблицей, созданной с использованием разработанного в рамках настоящей работы расширения – поиске строки в таблице, вставке строки в таблицу, обновлении строки в таблице, удалении строки из таблицы
- Производит перестраивание индексирующей структуры данных только на каждой 1000-й операции и только для первых 10000 операций
- Выбор индексирующей структуры данных зависит от соотношений количеств операций разных типов (поиск, вставка, удаление) с деревом
- B-дерево и его модификации, используемые в программном продукте, разработанном в рамках настоящей работы, имеют порядок 750
- По умолчанию в расширении при создании таблицы используется B^+-дерево

АЛГОРИТМ ВЫБОРА ИНДЕКСИРУЮЩЕЙ СТРУКТУРЫ ДАННЫХ

1. Если текущее общее количество операций с деревом равно 0, или больше 10000, или не кратно 1000, то выйти из алгоритма, иначе перейти к шагу 2.

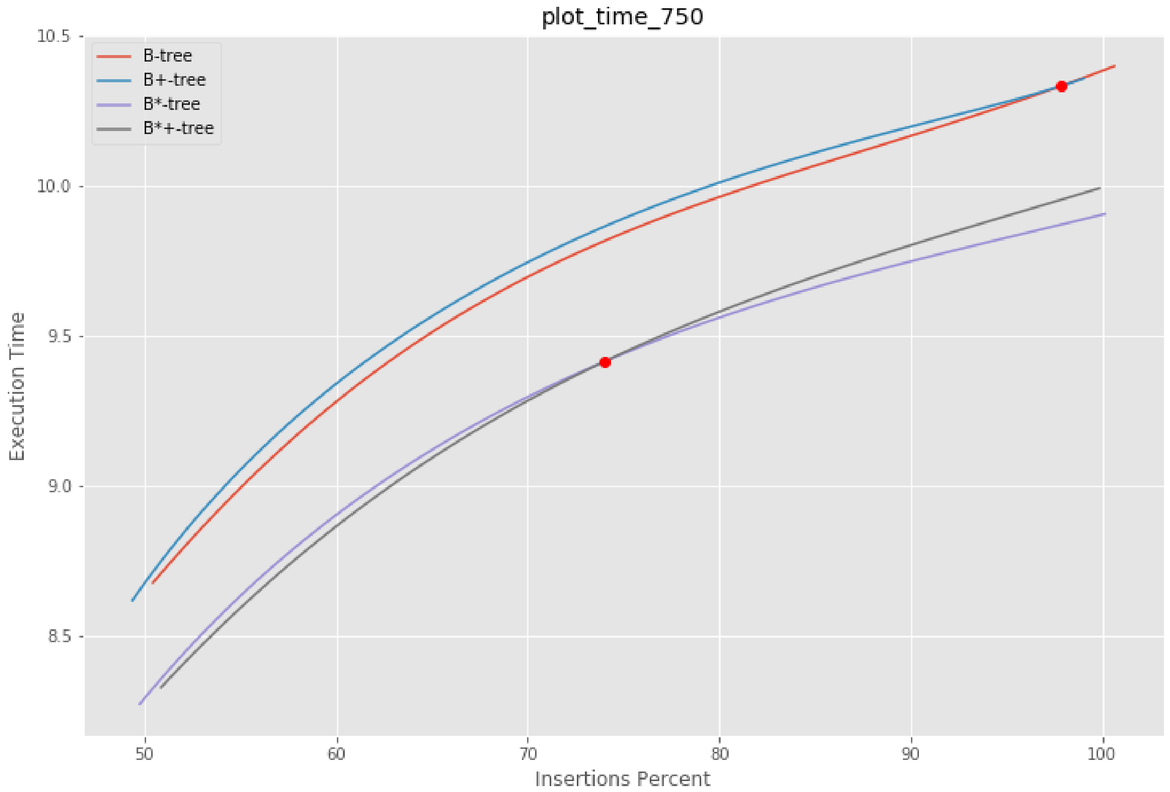
2. Если текущее количество операций с изменением данных в дереве (вставок ключей в дерево, удалений ключей из дерева) составляет менее 10 % от текущего общего количества операций с деревом, то выйти из алгоритма, иначе перейти к шагу 3.

3. Если текущее количество операций вставки в дерево составляет более 73,97 % от текущего количества операций с изменением данных в дереве, то выбрать в качестве индексирующей структуры данных B^*-дерево и перейти к шагу 5, иначе перейти к шагу 4.

4. Выбрать в качестве индексирующей структуры данных B^{*+}-дерево и перейти к шагу 5.

5. Если в шагах 3 – 4 была выбрана новая индексирующая структура данных, то перестроить имеющуюся индексирующую структуру данных на выбранную в шагах 3 – 4, сохранив все имеющиеся в ней данные.

АЛГОРИТМ ВЫБОРА ИНДЕКСИРУЮЩЕЙ СТРУКТУРЫ ДАННЫХ

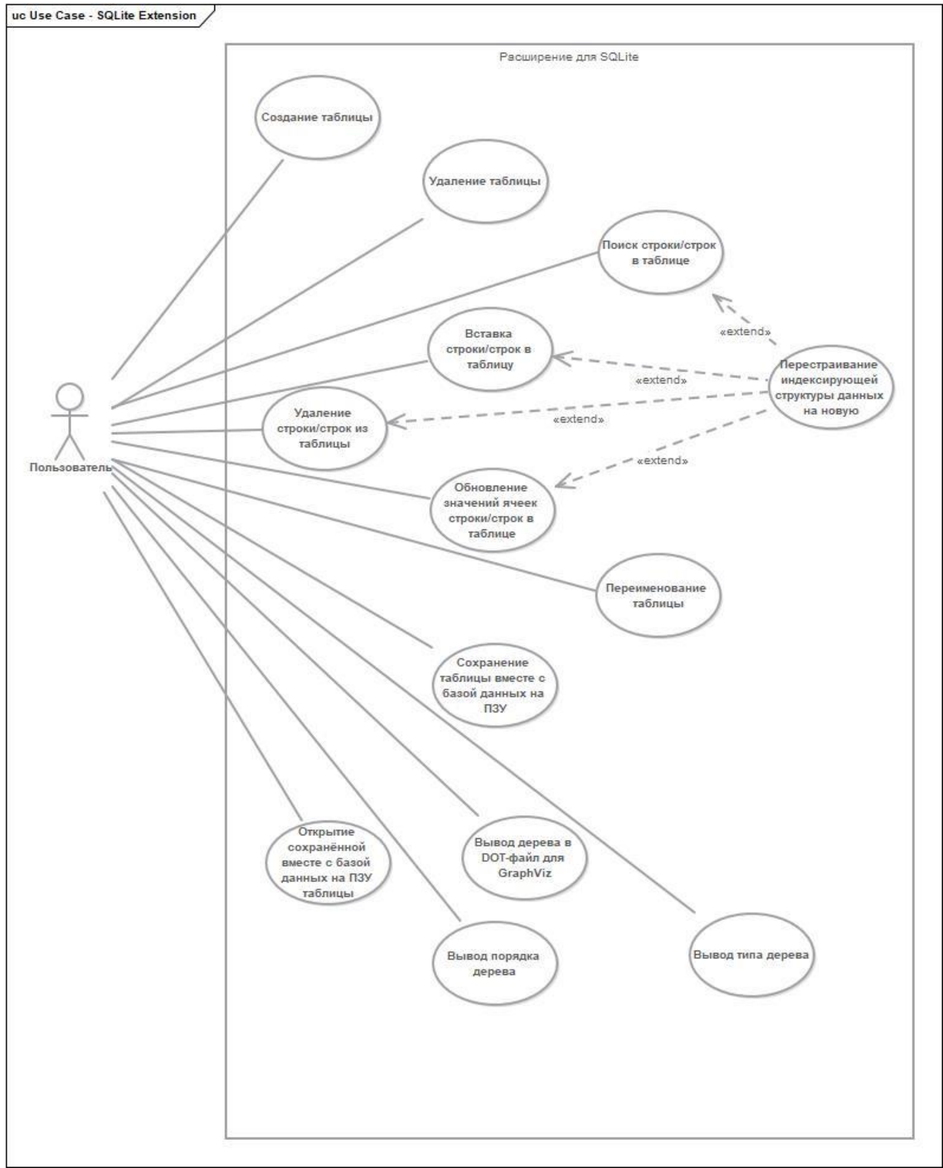


СТРОЕНИЕ КЛЮЧА В ДЕРЕВЕ

значение первичного ключа

row id (8 байт)

ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ



ТЕХНОЛОГИИ И ИНСТРУМЕНТЫ РЕАЛИЗАЦИИ

THE C PROGRAMMING LANGUAGE

GCC

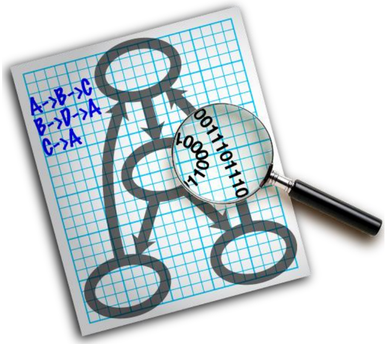
C++

SQLite

CL

python™

SQLite Studio



РЕАЛИЗАЦИЯ

- API на C реализовано с использованием конструкции *extern “C” { … }*
- Расширение для SQLite регистрирует в СУБД модуль виртуальной таблицы с названием *btrees_mods*, который «перехватывает» все обращения к виртуальным таблицам, созданным с использованием этого модуля
- Виртуальная таблица – любая таблица, созданная с использованием подобного модуля
- Также в СУБД при помощи расширения регистрируются функции *btreesModsVisualize*, *btreesModsGetTreeOrder*, *btreesModsGetTreeType*

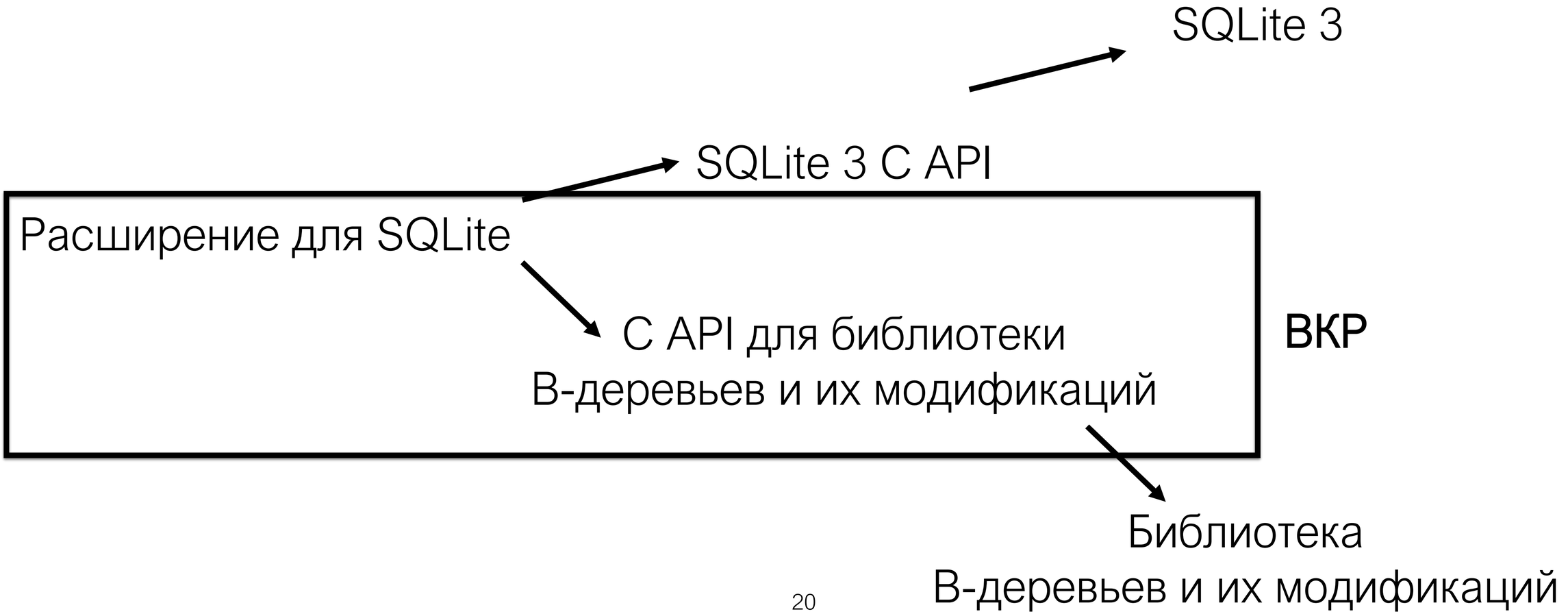
РЕАЛИЗАЦИЯ

Метод	Назначение
btreesModsCreate(sqlite3*, void*, int, const char* const*, sqlite3_vtab**, char**)	Создаёт новую таблицу
btreesModsUpdate(sqlite3_vtab*, int, sqlite3_value**, sqlite_int64*)	Выполняет вставку, обновление или удаление строки из таблицы.
btreesModsFilter(sqlite3_vtab_cursor*, int, const char*, int, sqlite3_value**)	Ищет строку в таблице.

РЕАЛИЗАЦИЯ

Метод	Назначение
btreesModsVisualize(sqlite3_context*, int, sqlite3_value**)	Выводит графическое представление индексирующей структуры (дерева) таблицы в DOT-файл для GraphViz.
btreesModsGetTreeOrder(sqlite3_context*, int, sqlite3_value**)	Выводит порядок дерева, используемого как индексирующая структура таблицы.
btreesModsGetTreeType(sqlite3_context*, int, sqlite3_value**)	Выводит тип дерева (1 – B-tree, 2 – B^+-tree, 3 – B^*-tree, 4 – B^{*+}-tree), используемого как индексирующая структура таблицы.

СХЕМА ПРОГРАММЫ



ПРИМЕР ИСПОЛЬЗОВАНИЯ

```
SQLite version 3.26.0 2018-12-01 12:34:55
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .load ./btrees_mods
sqlite> CREATE VIRTUAL TABLE btt USING btrees_mods(id INTEGER PRIMARY KEY, a INTEGER, b TEXT);
sqlite> INSERT INTO btt VALUES (4, 2, "ABC123");
sqlite> INSERT INTO btt VALUES (7, 3, "def");
sqlite> SELECT * FROM btt WHERE id = 4;
4|2|ABC123
sqlite> SELECT * FROM btt WHERE id = 7;
7|3|def
sqlite> SELECT * FROM btt WHERE id = 4 OR id = 7;
4|2|ABC123
7|3|def
sqlite> .tables
btrees_mods_idxinfo  btt                  btt_real
sqlite> SELECT * FROM btt_real;
4|2|ABC123
7|3|def
sqlite> SELECT * FROM btrees_mods_idxinfo;
btt|2|0|id|INTEGER|4|tree_33051558297088.btree
sqlite> DROP TABLE btt;
sqlite> .tables
btrees_mods_idxinfo
sqlite> SELECT * FROM btrees_mods_idxinfo;
sqlite> .exit
```

ЭКСПЕРИМЕНТ ПО СРАВНЕНИЮ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ ОПЕРАЦИЙ НА ДЕРЕВЬЯХ РАЗНЫХ ТИПОВ

Операция с таблицей	Общее время выполнения (мс)	Среднее время выполнения на одну строку (мс)
Создание таблицы	20	-
Вставка первых 500 строк	10301	20,6
Вставка последующих 500 строк	10322	20,6
Вставка 1001-й строки (включая перестраивание B^+-дерева в B^*-дерево)	40	40
Вставка последующих 499 строк	9386	18,8
Вставка последних 500 строк	9032	18,1

ЭКСПЕРИМЕНТ ПО СРАВНЕНИЮ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ ОПЕРАЦИЙ НА ДЕРЕВЬЯХ РАЗНЫХ ТИПОВ

Операция с таблицей	Общее время выполнения (мс)	Среднее время выполнения на одну строку (мс)
Удаление первых 500 строк	11558	23,1
Удаление последующих 500 строк	10708	21,4
Удаление 1001-й строки (включая перестраивание B^*-дерева в B^{*+}-дерево)	62	62
Удаление последующих 499 строк	9418	18,9
Удаление последних 500 строк	8863	17,7
Вставка 1000 строк	18890	18,9
Вставка последующих 5000 строк (включая перестраивание B^{*+}-дерева в B^*-дерево)	92395	18,5

ЭКСПЕРИМЕНТ ПО СРАВНЕНИЮ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ ОПЕРАЦИЙ НА ДЕРЕВЬЯХ РАЗНЫХ ТИПОВ

- На B^*-дереве вставка новых элементов в рамках данного эксперимента выполняется быстрее, чем на B^+-дереве
- На B^{*+}-дереве удаление в рамках данного эксперимента выполняется быстрее, чем на B^*-дереве
- Выигрыш в скорости вставки у B^*-дерева перед B^{*+}-деревом в рамках данного эксперимента незначителен
- Поиск строки в таблице занял, в среднем, 1 мс, на всех типах деревьев

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

- ✓ Реализовано API на C для имеющейся C++-библиотеки алгоритмов над сильно ветвящимися деревьями
- ✓ Разработано расширение для SQLite
- ✓ В расширении реализована функциональность для вывода графического представления используемого дерева и основной информации о нём
- ✓ Разработан и реализован алгоритм выбора структуры данных в качестве индекса
- ✓ Разработана техническая документация в соответствии с ЕСПД
- ✓ **Таким образом, все поставленные задачи выполнены**

АПРОБАЦИЯ РАБОТЫ

- Работа представлена на Студенческой конференции ФКН CoCoS’2019 в исследовательском треке
- Работа представлена на Международной конференции SYRCoSE 2019
- Статья по работе принята к публикации в Трудах ИСП РАН (Proceedings of ISP RAS) – № 1952 в Перечне ВАК

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

- Результаты работы могут быть использованы разработчиками и исследователями, для сравнения параметров эффективности модификаций B-дерева и использования модификаций B-дерева в качестве индексирующих структур данных в SQLite, в том числе, в учебных и научных целях
- Направления дальнейших разработок:
  - <, <=, >, >= при поиске строк по первичному ключу
  - Поддержка транзакционности
  - Поддержка команд с JOIN
  - Разработка плагина для одного из SQLite-менеджеров с графическим пользовательским интерфейсом, для более удобной работы с B-деревьями и их модификациями
  - Доработки C++-библиотеки сильно ветвящихся деревьев, для снижения сложности операций

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Comer D. The Ubiquitous B-Tree // ACM Computing Surveys. – 1979. – June (vol. 11, no. 2). – P. 121 – 137.

2. Pollari-Malmi K. B+-trees // [Электронный ресурс]: Computer Science | University of Helsinki. Режим доступа: https://www.cs.helsinki.fi/u/mluukkai/tirak2010/B-tree.pdf, свободный. (дата обращения: 07.12.2017).

3. Run-Time Loadable Extensions // [Электронный ресурс]: SQLite. Режим доступа: https://www.sqlite.org/loadext.html, свободный (дата обращения: 04.11.2018)

4. Кормен Т. Алгоритмы: построение и анализ. 3-е изд. / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. — М.: ИД «Вильямс». — 2013. — 1324 с.

5. Ригин А.М. Исследование эффективности сильно ветвящихся деревьев в задаче индексирования структурированных данных : Курсовая работа / Ригин Антон Михайлович; НИУ ВШЭ. – М., 2018. – 37 с.

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Спасибо за внимание!

amrigin@edu.hse.ru
anton19979@yandex.ru
anton19979@yandex-team.ru