МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

СТРУКТУРЫ ДАННЫХ «КЛЮЧ-ЗНАЧЕНИЕ» И ИХ ПРИМЕНЕНИЕ

Учебно-методическое пособие

Воронеж
Издательский дом ВГУ

2018

Содержание

Помимо различия в моделях данных, NoSQL СУБД сильно отличаются друг от друга и внутренним устройством. Для реализации хранилищ используются различные структуры данных: хеш-таблицы, деревья и другие, которые делятся на ассоциативные («ключ-значение») и последовательные. Последовательные контейнеры хранят данные в памяти линейно, сохраняя относительные позиции элементов. Ассоциативные контейнеры не сохраняют порядок, в котором вставляются элементы, а вместо этого используют ключи, хранящиеся в элементе (или сам элемент в качестве ключа) для достижения максимальной скорости вставки, выборки и удаления [13]. В данном пособии будет рассматриваться построение именно ассоциативных контейнеров, так как они являются наиболее популярными и используемыми в NoSQL [15] базах данных.

Методы построения контейнеров данных «ключ-значение» можно разделить на две категории. Одна группа методов подразумевает использование некоторого глобального упорядочения (числового или лексикографического). Ключи хранятся в отсортированном состоянии, а для поиска используется бинарный алгоритм. Контейнеры, полученные данным методом, называются отсортированными ассоциативными контейнерами. Примерами таких контейнеров являются различные деревья.

Второй группой методов является хеширование, а контейнеры, полученные данным методом, называются хешированными ассоциативными контейнерами. Примерами таких контейнеров являются различные вариации хеш-таблиц.

На рис. 1 изображена классификация ассоциативных контейнеров, проанализированных в данном пособии.

2. Отсортированные ассоциативные контейнеры

В отсортированном ассоциативном контейнере все ключи отсортированы в некотором порядке. Простейшим примером такого контейнера является Sorted String Table (SSTable) [40]. Этот контейнер является одним из

самых популярных для хранения, обработки и обмена большими наборами данных. Он используется в таких известных NoSQL базах данных, как Cassandra [28], HBase и LevelDB. Данный контейнер хорошо подходит для ситуаций, когда необходимо обработать гигабайты или даже терабайты информации и выполнить на них последовательность из заданий Map-Reduce [7]. В этом случае из-за размера входных данных затраты на операции чтения и записи будут преобладать над затратами на выполнение полезной работы.

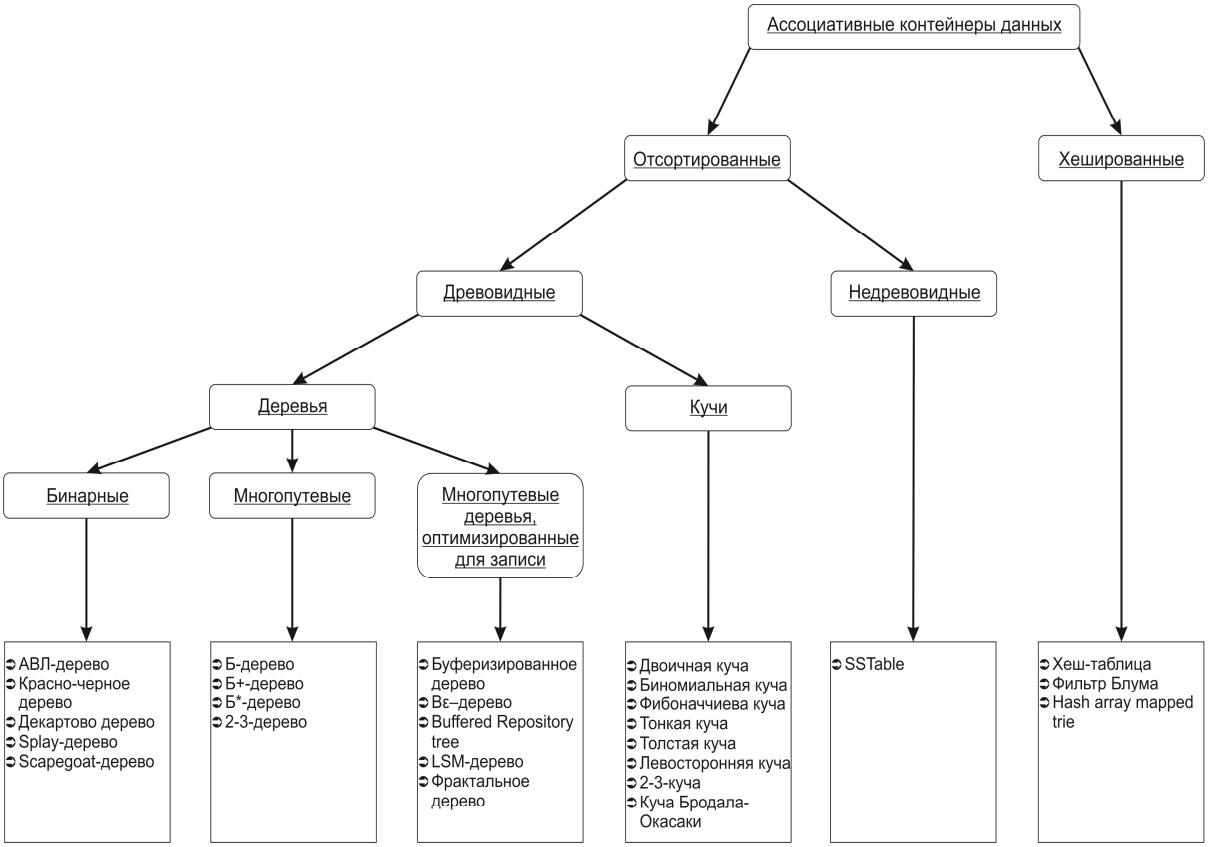

Рис. 1

Таким образом, использование случайного чтения/записи крайне неэффективно; вместо этого можно использовать потоковое чтение и запись, что позволяет значительно уменьшить расходы на операции чтения/записи. SSTable позволяет использовать такое чтение за счет своего внутреннего устройства. Данный контейнер представляет собой последовательность от-

сортированных по ключу пар «ключ-значение». Следовательно, временная сложность основных операций над этим контейнером соответствует временной сложности отсортированного массива (табл. 1).

Таблица 1

Алгоритм	Среднее	Худший случай
Место	O(n)	O(n)
Поиск	O(log n)	O(log n)
Вставка	O(n)	O(n)
Удаление	O(n)	O(n)

SSTable позволяет хранить дубликаты и не имеет ограничений на размер ключей и значений. Кроме того, для данного контейнера можно обеспечить доступ по ключу за константное время. Для этого необходимо создать отсортированный индекс, последовательно считывая данные из контейнера. Индекс будет упорядочен по ключу и содержит смещения до соответствующих значений, как показано на рис. 2.

Главное преимущество SSTable состоит в простоте и эффективности хранения и обработки большого объема данных, но у данного контейнера существует и значительный минус. Для работы с большими данными необходимо, чтобы SSTable размещался на жестком диске. Следовательно, любое изменение контейнера (удаление или вставка элемента) будет требовать больших затрат на дисковый ввод-вывод. Это связано с тем, что SSTable представляет собой отсортированный по ключу массив и для его изменения необходимо перестроить весь контейнер.

Таким образом, SSTable лучше всего использовать для хранения статических индексов, так как для извлечения с диска можно последовательно считывать данные, установив головку диска всего один раз. Достаточно бы-

стро происходит и случайное чтение. Следовательно, любая операция по изменению данного контейнера является слишком длительной и дорогостоящей, особенно когда SSTable располагается не в оперативной памяти. Поэтому обычно в NoSQL СУБД SSTable, которые располагаются на диске, являются неизменяемыми.

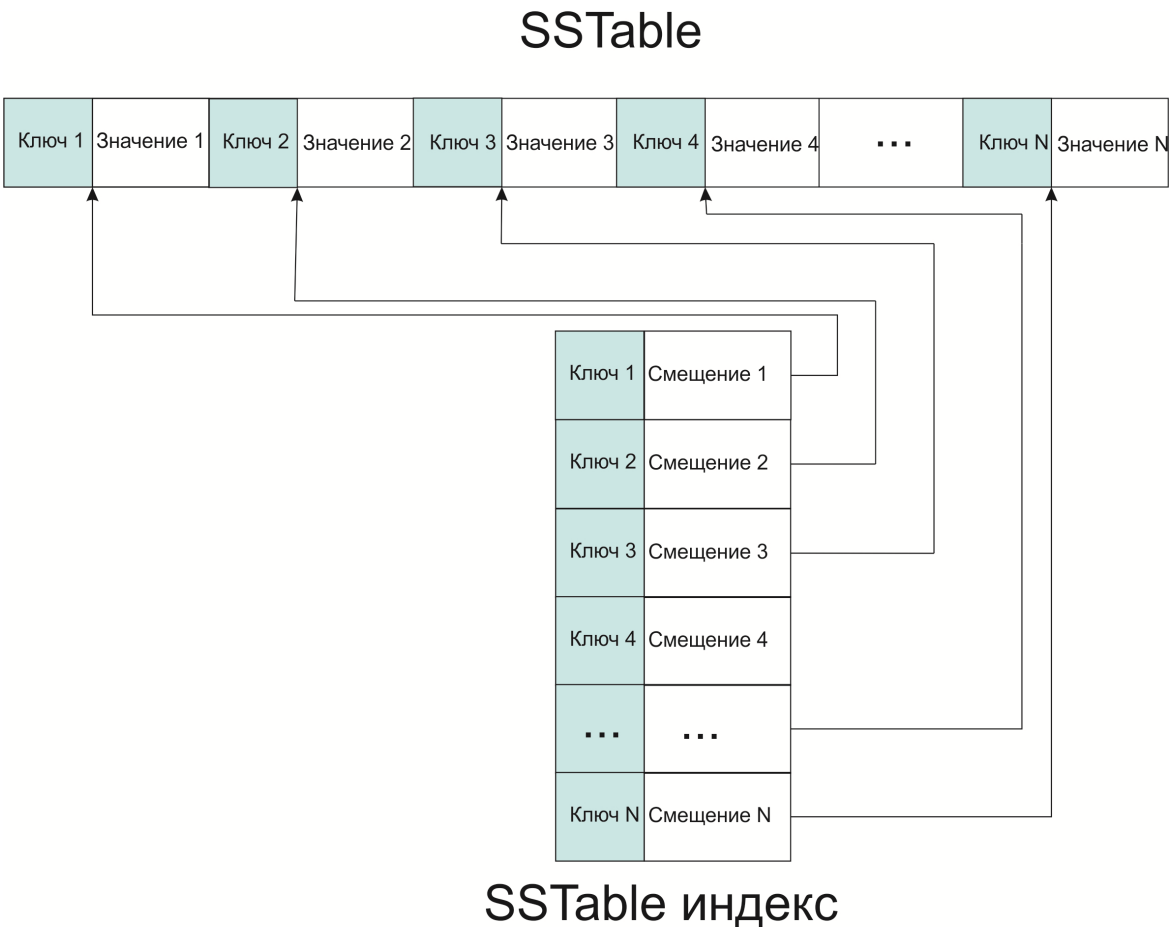

Рис. 2

Для решения основной проблемы SSTable (крайне неэффективной случайной записи) был разработан другой контейнер – LSM (Log-Structured Merge) дерево [41], которое будет описано ниже.

При использовании данного контейнера в самоадаптирующихся контейнерах [5] необходимо учитывать, что перестроение в SSTable из других контейнеров в основном требует больших затрат (особенно если SSTable располагается на диске), так как данные в памяти или на диске должны

быть упорядочены по ключу. Данный контейнер следует использовать в самоадаптирующихся контейнерах только тогда, когда известно, что длительное время контейнер будет использоваться только для частого чтения больших данных, а следовательно, преимущества последовательного чтения перекроют затраты на построение.

Помимо SSTable существует другая группа отсортированных ассоциативных контейнеров – деревья поиска.

3. Древовидные структуры

Древовидная структура [2] данных – это динамическая структура, в которой связи между элементами не линейны как в списке, а похожи на ветви дерева. Существует две категории данных структур, которые различны по методам построения и обработки. Первая – это *деревья*, вторая – *кучи*. Кроме того, деревья различают по другим атрибутам.

1. Сбалансированность. Дерево может быть:
   - вырожденным;
   - идеально сбалансированным;
   - сбалансированным;
   - несбалансированным и невырожденным.
2. Количество ветвей. Дерево может быть:
   - двоичным;
   - многопутевым, когда количество ветвей больше двух.

4. Бинарные деревья

Бинарное дерево поиска – это бинарное дерево [10], соответствующее следующим критериям:

- потомками каждого узла X являются два поддерева, которые тоже являются бинарными деревьями поиска;
- элементы левой ветви некоторой вершины X имеют значения меньше, чем значение вершины X;