Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Классификация методов построения индексов в базах данных»

Студент: ИУ7-73Б _____ М. Д. Маслова
(группа) (подпись, дата) (И. О. Фамилия)

Преподаватель: _____ А. А. Оленев
(подпись, дата) (И. О. Фамилия)

2022 г.

РЕФЕРАТ

Расчетно-пояснительная записка 12 с., 0 рис., 0 табл., 0 источн., 1 прил.

Ключевые слова:

Краткое описание

Мои заметки

- определение индекса;
- проблемы при построении индексов (доп память, затраты на изменение дерева при изменении базы данных и тп);
- реляционные/нереляционные бд???;
- типы индексов:
    - B-tree
    - Hash
    - Bitmap
- обученные индексы (learned indexes);
- критерии:
    - затраты на перестроение (добавление/удаление);
    - увеличение скорости;
    - ???

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

АКТУАЛЬНОСТЬ РАБОТЫ

Целью данной работы является ***классификация методов построения индексов в базах данных***.

Для достижения поставленной цели требуется решить следующие задачи:

- описываются методы построения индексов в базах данных;
- предлагаются и обосновываются критерии оценки качества описанных методов;
- сравниваются методы по предложенным критериям оценки;
- выделяются методы, показывающие лучшие результаты по одному или нескольким критериям.

1 Анализ предметной области

Данных много => актуально => базы данных.

База данных — это ...

Основная операция — поиск => создание методов для ускорения данной операции, одним из которых является индексы (***есть ли другие???***).

Индекс – это ...

Существует два основных вида индексов ***уточнить???***:

- упорядоченные, реализующиеся на основе деревьев поиска;
- хеш-индексы, в которых поиск значений осуществляется с помощью вычисления хеш-функции.
- ***bitmap-индексы??? (индексы на основе битовых карт).***

Индекс представляет собой структуру, которая строится в дополнение к существующим данным. Таким образом, она занимает дополнительный объем памяти и должна соответствовать текущим данным, то есть необходимо изменять данную структуру при вставке или удалении элементов. Так как индексы создаются для осуществления поиска, то они также характеризуются типом и временем доступа.

Таким образом, можно выделить следующие характеристики индексов (***мб по ним и оценивать, почему нет***):

- ***тип доступа*** — поиск записей по аттрибуту с конкретным значением, или со значением из указанного диапазона;
- ***время доступа*** — время поиска записи или записей;
- ***время вставки***, включающее время поиска правильного места вставки, а также время для обновления индекса;
- ***время удаления***, аналогично вставке, включающее время на поиск удаляемого элемента и время для обновления индекса;
- ***дополнительная память***, занимаемая индексной стркутурой.

Ключ поиска — аттрибут или набор аттрибутов, по которым осуществляется поиск записей.

2 Описание существующих решений

2.1 Последовательные индексы

Последовательные индексы деляется на ***кластеризованные (или первичные) и некластеризованные (или вторичные)???***. В первых записи в проиндексированном файле храняется ***последовательно упорядоченно (что это значит???)***, во вторых в порядке отличном от последовательного.

рисунки примеры

Также индексы деляется на разреженные и неразреженные.

Неразреженные индексы содержат ключ поиска и указатель на первую запись с заданным ключом поиска. При этом в первичных индексах другие записи с заданным ключом будут лежать сразу после первой записи, так как записи в таких файлах отсортированны по тому же ключу. Неразреженные вторичные индексы должны содержать список указателей на каждую запись с заданным ключом поиска.

(Сортировать записи в файле??? Не выгодно, только если основная часть запросов не осуществляется по этому индексу, и сортировка дешевле множества поисков)

В ***разреженных индексах*** записи содержат только некоторые значения ключа поиска, а для доступа к элементу отношения ищется запись индекса с наибольшим меньшим или равным значением ключа поиска, происходит переход по указателю на первую запись по найденному ключу и далее по указателям в файле происходит поиск заданной записи. Таким образом, разреженные индексы могут быть построены только на отсортированных последовательностях записей, иначе хранения только некоторых ключей поиска будет недостаточно, так как будет неизвестно, после записи, с каким ключом будет лежать необходимый элемент отношения.

Рисунки как в книжке

Поиск с помощью неразреженных индексов быстрее, так как указатель в записи индекса сразу приводит к необходимым записям. Однако разреженные индексы требуют меньше дополнительной памяти и сокращают время поддержания структуры индекса в актуальном состоянии при вставке или удалении.

Решение, о том какой тип индексов использовать, принимается разработ-

чиком для каждого приложения отдельно. ***(Хорошо — разреженный индекс по блокам памяти, так как основные затраты по времени приходятся на загрузку данных из вторичной памяти в оперативную, но хорошо — абстрактное понятие; + как решить ситуация, когда данные одного ключа поиска лежат в разных блоках ;)***.

2.1.1 B-tree индексы

B-tree индексы можно рассматривать как модель сопоставления ключа позиции искомой записи в отсортированном массиве.

Лучший выбор для запросов диапазонов в силу упорядоченности данных.

Такие индексы как бы предсказывают положение записи с ***минимаксной (верно?)*** ошибкой (min err = 0, max err = page_size). Поэтому можем заминить B-деревья на линейную модель также с минимаксной ошибкой (возможно/скорее всего другой).

Так для предсказания можно представлять Range Index Models как модели функции распределения:

$$\text{position} = F(key) \cdot N, \tag{2.1}$$

где $F(key)$ — функция распределения, дающая оценку вероятности обнаружения ключа, меньшего или равного ключу поиска, то есть $P(X < key)$;

N — количество ключей.

График позиции от ключа

Можно построить индексы на основе рекурсивной модели, в которой строится иерархия моделей из n уровней (этапов). Каждая модель на вход получает ключ, на основе которого выбирает модель на следующем уровне. Модели последнего этапа предсказывают положение записи.

Картиночка иерархии моделей.

Можно использовать различные модели: например, на верхнем использовать нейнонные сети, а на нижних простые линейные регрессионные модели или даже простые B-деревья.

Алгоритм страница 8.

Для индексирвония строк используют токенизацию — представление строки в виде чисел (NLP).

2.1.2 B^+-tree индексы

2.1.3 R-tree индексы (не из книги, часто упоминаются в статьях)

2.1.4 LSM-tree индексы (оптимизация)

2.1.5 Буферные деревья

Нужно написать, что есть куча других деревьев, описаны самые часто используемые ;)

2.2 Хеш-индексы

Хеш-индексы можно рассматривать как модель сопоставления ключа позиции искомой записи в неупорядоченном массиве.

Лучшее — поиск по простому ключу.

Фукнция распределения вероятностей распределения ключей (CDF of the key detribution) один из возможных способов обучения хеш-индексов. CDF масштабируется на размер хеш-таблицы M и для поиска положения записи аналогично случаю с B-деревьями используется формула:

$$h(K) = F(K) \cdot M, \tag{2.2}$$

где K — ключ.

2.3 Индексы на основе битовых карт

Данные индексы можно рассматривать как модель проверки существования записи в массиве данных.

Фильтр Блума? (как раз алгоритм используемый для проверки) очевидно эффективен для проверки существования записи.

Фильтр Блума использует массив бит размером m и k хеш-функций, каждая из которых сопоставляет ключ с одну из m позиций. Для добавления элемента в множество существующих значений ключ подается на вход каждой хеш-функции, возвращающих позицию бита, который должен быть установлен в 1. Для проверки принадлежности ключа множеству, ключ также подается на вход k хеш-функций. Если какой-либо бит, соответствующий одной из

возвращенных позиций, равен нулю, то ключ не входит во множество.Из этого следует, что данный алгоритм гарантирует отсутсвие ложноотрицательных результатов.

Может со 100%-ной вероятностью сказать, что элемент отсутсвует в наборе, но то, что элемент присутсвует в наборе, со 100%-ной вероятностью он сказать не может (возможны ложноположитеьные результаты)

В случае индексов существования необходимо обучить функцию таким образом, чтобы среди возвращенных значений для множества ключей были коллизии, аналогично для множества неключей, но при этом не было коллизий возвращенных значений для ключей и неключей. (это надо переписать, непонятно написано: возвращенные значения для ключей должны попадать в одни значения, для неключей – в другие, но множества возвращенных значений для ключей и неключей не должны совпадать)

В отличие от оригинального фильтра Блума, где FNR = 0, FPR = const, где const выбрано априори, при обучении достигается заданное значение FPR при FNR = 0 на реальных запросах.

2.4 Индексирование пространственных и временных данных

3 Классификация существующих решений

ЗАКЛЮЧЕНИЕ

Подвести к обученным индексам (learned indices) даже раньше заключения.

Бла-бла-бла — для поиска наилучших характеристик индексов можно использовать методы машинного обучения.

Мои предположения:

- достаточно иметь не полностью сбалансированное дерево поиска и при этом не проигрывать во времени доступа, но уменьшать потери при вставке и удалении);
- выбор с помощью ML структуры индекса (разреженный/неразреженный, кластеризованный/некластеризованный);
- простая замена всего индекса на предсказывающую положение записи обученную модель???