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 16 с., 9 рис., 0 табл., 7 источн., 1 прил.

Ключевые слова:

Краткое описание

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На протяжении последнего десятилетия происходит автоматизация все большего числа сфер человеческой деятельности [1]. Это приводит к тому, что с каждым годом производится все больше данных. Так, по исследованию компании IDC (International Data Corporation), занимающейся изучением мирового рынка информационных технологий и тенденций развития технологий, объем данных к 2025 году составит около 175 зеттабайт, в то время как на год исследования их объем составлял 33 зеттабайта [2]. При этом данные необходимо хранить и обрабатывать.

Хранятся в базе данных. Поступают запросы, которые обрабатывает СУБД. Поиск. Уменьшение времени обработки запроса -> несколько методов -> один из них индексы. 1.

Поэтому проводятся исследования для уменьшения временных и пространственных сложностей. Так, в 2018 году авторами статьи [3] было проведено исследование...

Почему актуальны индексы?

Большие данные -> много запросов -> необходимость быстрого поиска -> индексы.

Машинное обучение в индексах.

В 2018 году статья Learned Indexes -> показали уменьшение времени поиска -> заинтерисованность научного сообщества -> много статей [**1**][**2**][**3**]...

Так как Learned Indexes используют идеи базовых индексных структур: деревья поиска, хеш-таблицы, битовые карты.

Поэтому в этой работе для каждой из вышеперечисленных структур сначала описывается построения индекса на основе ее, а далее приводится описание соответсвтующего обученного индекса.

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

1.1 Основные определения

Индекс — это некоторая структура, обеспечивающая быстрый поиск записей в базе данных [4]. Индекс определяет соответствие значения аттрибута или набора аттрибутов — *ключа поиска* — конкретной записи с местоположением этой записи [5]. Это соответствие организуется с помощью индексных записей. Каждая из них соответствует записи в *индексируемой таблице* — таблице, по которой строится индекс — и содержит два поля: идентификатор записи или указатель на нее, а также значение индексированного поля в этой записи [6].

Индексы могут использоваться для поиска по конкретному значению или диапазону значений, а также для проверки существования элемента в таблице, однако обеспечение уменьшения времени доступа к записям в общем случае достигается за счет [5]:

- упорядочивания индексных записей по ключу поиска, что уменьшает количество записей, которые необходимо просмотреть;
- а также меньшего размера индекса по сравнению с индексируемой таблицей, сокращающего время чтения одного элемента.

В то же время индекс является структурой, которая строится в дополнение к существующим данным, то есть он занимает дополнительный объем памяти и должен соответствовать текущим данным. Последнее значит, что индекс необходимо изменять при вставке или удалении элементов, на что затрачивается время, поэтому индекс, ускоряя работу СУБД при доступе к данным, замедляет операции изменения таблицы, что необходимо учитывать[7].

Таким образом, индекс может описываться: [5]:

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

1.2 Типы индексов

Индексы могут быть:

- кластеризованные и некластеризованные;
- плотные и разреженные;
- одноуровневые и многоуровневые;
- а также иметь в своей основе различные структуры, что описывается в следующем разделе, так как исследуется в данной работе.

В *кластеризованных* индексах логический порядок ключей определяет физическое расположение записей, а так как строки в таблице могут быть упорядочены только в одном порядке, то кластеризованный индекс может быть только один на таблицу. Логический порядок *некластеризованных* индексов не влияет на физический, и индекс содержит указатели на записи таблицы.

Плотные индексы (рисунок 1.1) содержат ключ поиска и указатель на первую запись с заданным ключом поиска. При этом в кластеризованных индексах другие записи с заданным ключом будут лежать сразу после первой записи, так как записи в таких файлах отсортированны по тому же ключу. Плотные некластеризованные индексы должны содержать список указателей на каждую запись с заданным ключом поиска.

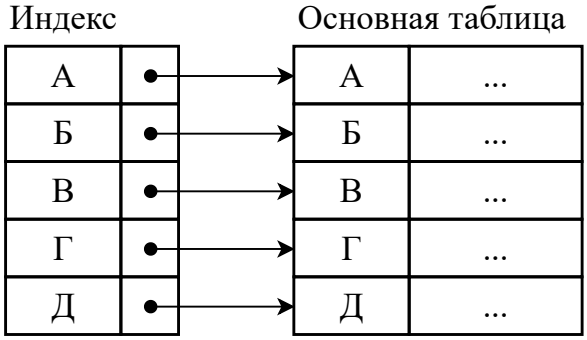

Рисунок 1.1 – Плотный индекс

В *разреженных* индексах (рисунок 1.2) 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.

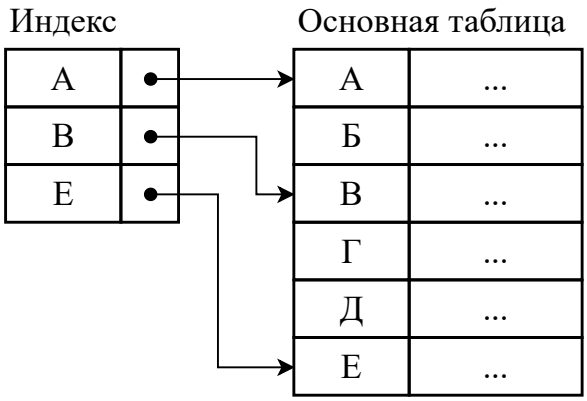

Рисунок 1.2 – Разреженный индекс

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.

Одноуровневые индексы ссылаются на данные таблице, индексы же верхнего уровня *многоуровневой* струкуры ссылают на индексы нижестоящего уровня (рисунок 1.3).

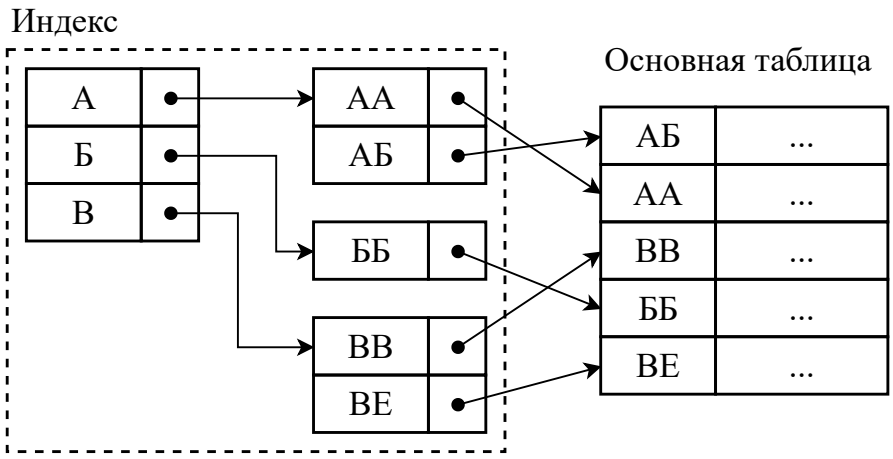

Рисунок 1.3 – Многоуровневый индекс

2 Описание существующих методов построения индексов

По структуре индексы подразделяются на

- упорядоченные, на основе деревьев поиска,
- индексы на основе хеш-таблиц,
- индексы на основе битовых карт.

2.1 Индексы на основе деревьев поиска

B-tree индексы можно рассматривать как модель сопоставления ключа позиции искомой записи в отсортированном массиве (рисунок 2.1).

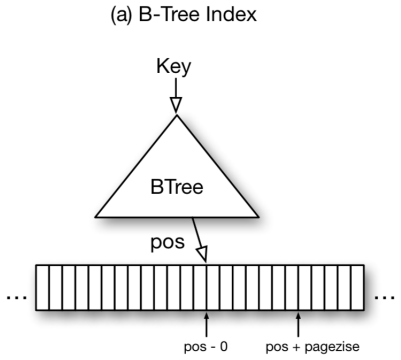

Рисунок 2.1 – B-деревья

Такие индексы как бы предсказывают положение записи с минимаксной ошибкой ($min\_err = 0$, $max\_err = page\_size$). Поэтому можем заминить B-деревья на линейную модель также с минимаксной ошибкой (рисунок 2.2).

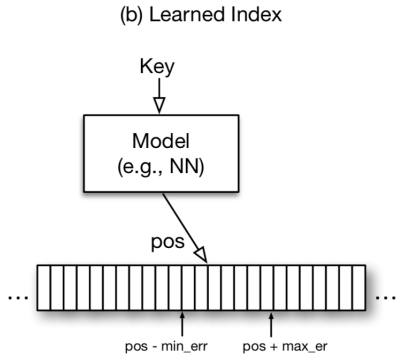

Рисунок 2.2 – Обученный индекс

Так для предсказания можно представлять Range Index Models как модели функции распределения (рисунок 2.3):

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

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

N — количество ключей.

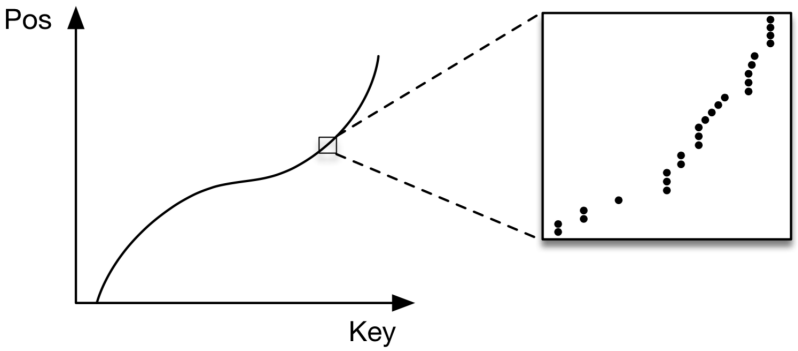

Рисунок 2.3 – Индекс как функция распределения

Можно построить индексы на основе рекурсивной модели (рисунок 2.4), в которой строится иерархия моделей из n уровней. Каждая модель на вход получает ключ, на основе которого выбирает модель на следующем уровне. Модели последнего этапа предсказывают положение записи.

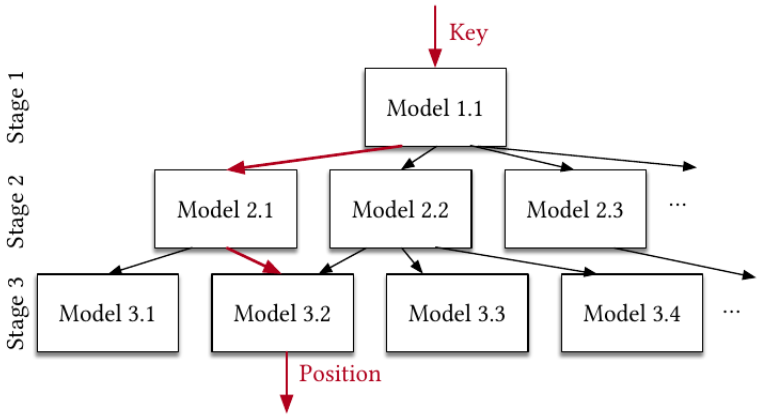

Рисунок 2.4 – Рекурсивная модель индекса

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

2.2 Индексы на основе хеш-таблиц

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

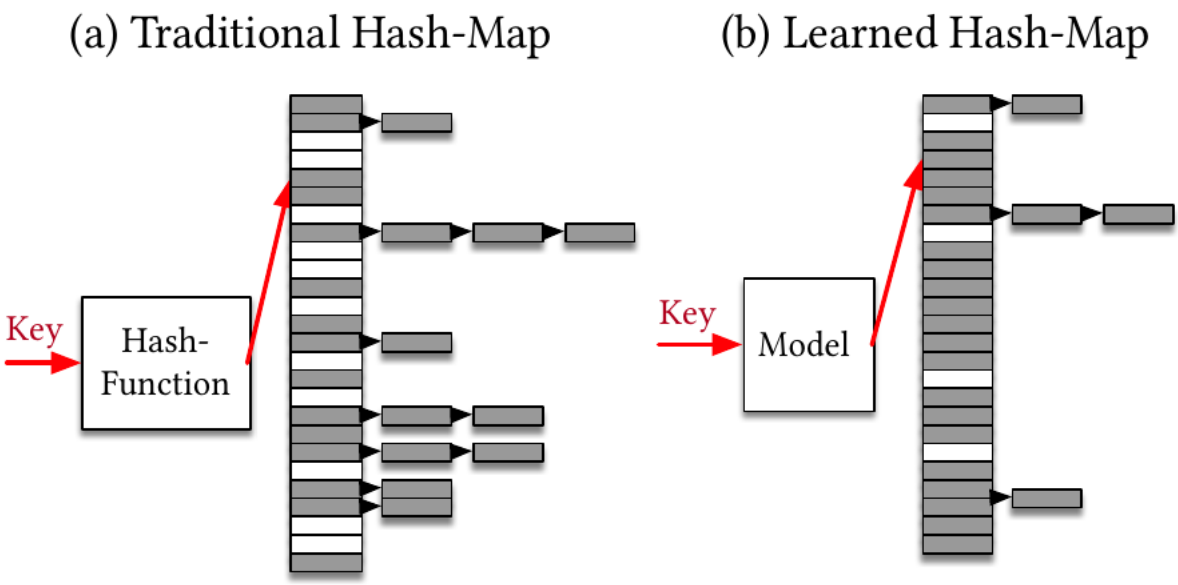

Рисунок 2.5 – Хеш-индексы

Функция распределения вероятностей распределения ключей один из возможных способов обучения хеш-индексов. Функция распределения 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 — алгоритм используемый 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 единицу. 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.

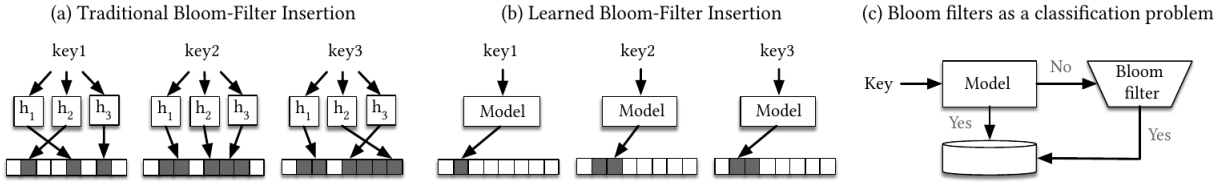

Рисунок 2.6 – Bitmap-индексы

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.

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.

3 Классификация существующих решений

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Исследование способов ускорения поисковых запросов в базах данных / Е. В. Коптенок, М. А. Подвесовская, Т. М. Хвостенко, А. В. Кузин // Вестник образовательного консорциума среднерусский университет. Информационные технологии. — 2019. — N 1(13). — С. 24—27.
2. *Reinsel D.*, *Gantz J.*, *Rydning J.* The Digitization of the World From Edge to Core // IDC White Paper. — 2018.
3. The Case for Learned Index Structures / T. Kraska [et al.] // Proceedings of the 2018 International Conference on Management of Data. — SIGMOD'18, June 10–15, 2018, Houston, TX, USA, 2018. — P. 489–504.
4. *Григорьев Ю. А.*, *Плутенко А. Д.*, *Плужникова О. Ю.* Реляционные базы данных и системы NoSQL: учебное пособие. — Благовещенск : Амурский гос. ун-т, 2018. — 424 с.
5. *Silberschatz A.*, *Korth H. F.*, *Sudarshan S.* Database System Concepts. — New York : McGraw-Hill, 2020. — 1344 p.
6. *Эдвард Сьоре*. Проектирование и реализация систем управления базами данных. — М. : ДМК Пресс, 2021. — 466 с.
7. *Осипов Д. Л.* Технологии проектирования баз данных. — М. : ДМК Пресс, 2019. — 498 с.