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., 8 рис., 0 табл., 4 источн., 1 прил.

Ключевые слова:

Краткое описание

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

1.1 Основные определения

Индекс — это некоторая структура, обеспечивающая быстрый поиск записей в базе данных [1]. Индекс определяет соответствие значения аттрибута или набора аттрибутов — *ключа поиска* — конкретной записи с местоположением этой записи [2]. 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 [3].

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 [2]:

- 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[4].

Таким образом, индекс может описываться: [2]:

- *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:

- первичные и вторичные (???);
- 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.

первичные — по первичному ключу/по уникальным значениям/=кластеризованные,

вторичные — по всем остальным аттрибутам/по неукальным значениям/= некластеризованные;

По порядку записей в индексируемой таблице индексы делятся на кластеризованные и некластеризованные. 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.

Также индексы делятся на плотные и разрженные. *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.

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

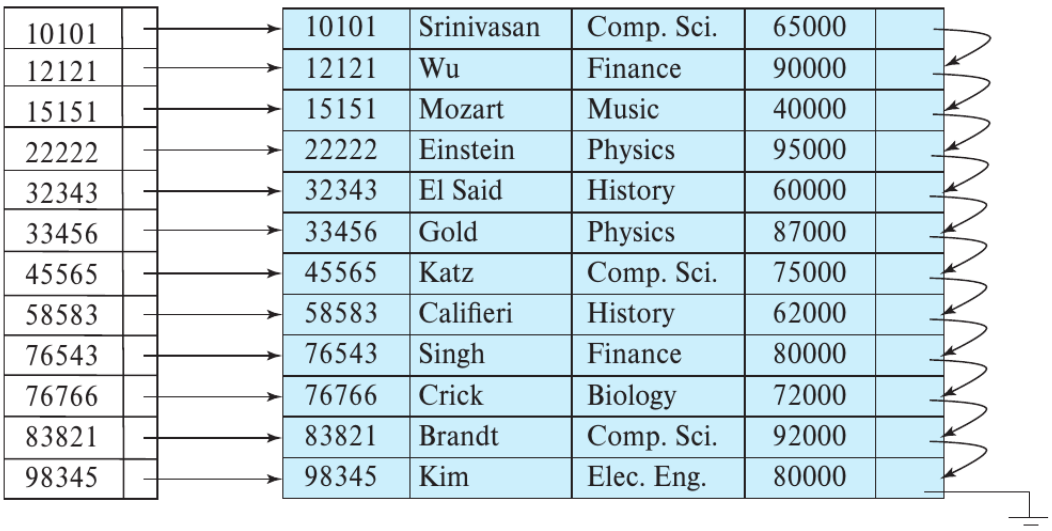

Рисунок 1.1 – Плотный индекс

будет недостаточно, так как будет неизвестно, после записи, с каким ключом будет лежать необходимый элемент отношения.

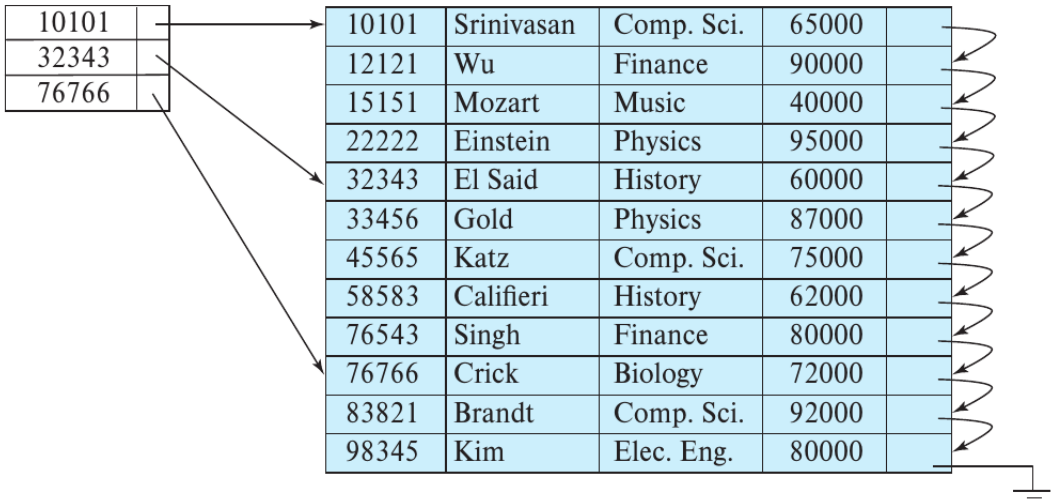

Рисунок 1.2 – Разреженный индекс

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).

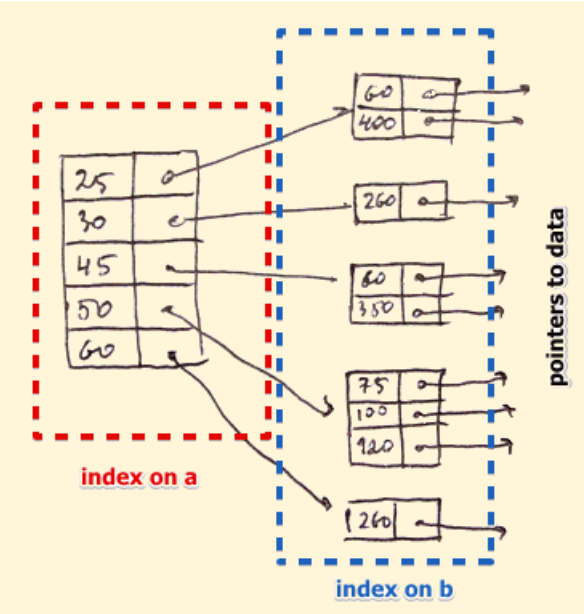

Рисунок 1.3 – Многоуровневый индекс

По структуре индексы подразделяются на

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

Построение структур каждого из приведенных типов индекса рассматривается в отдельном разделе, так как именно оно исследуется в данной работе.

2 Описание существующих решений

2.1 Индексы на основе деревьев поиска

2.1.1 B-деревья

2.1.1.1 Из статьи

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

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 (возможно/скорее всего другой).

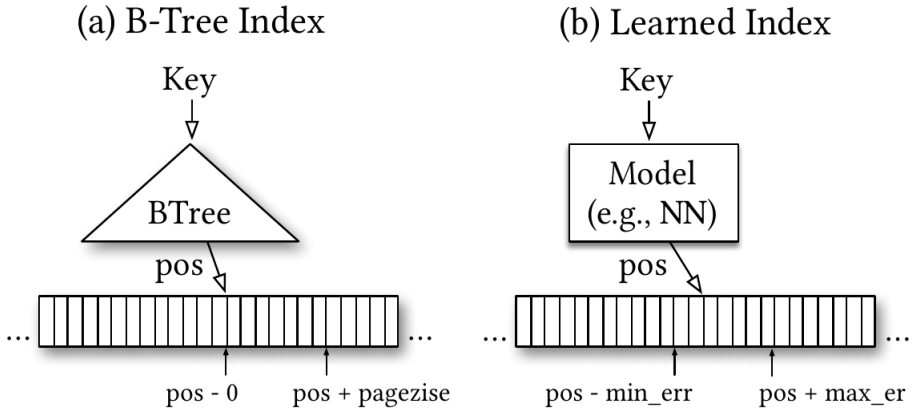

Рисунок 2.1 – B-деревья

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.

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.

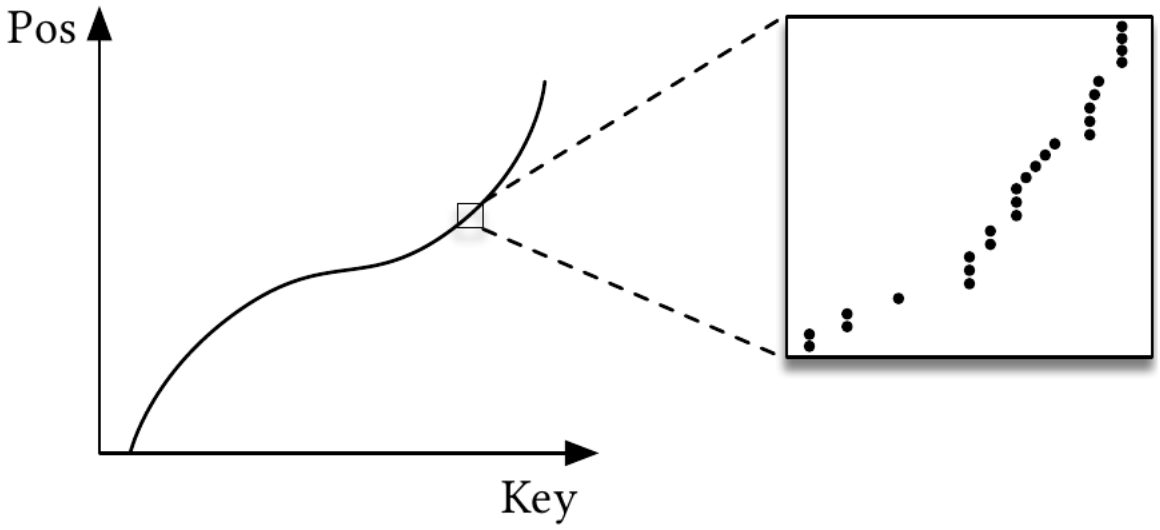

Рисунок 2.2 – Индекс как функция распределения

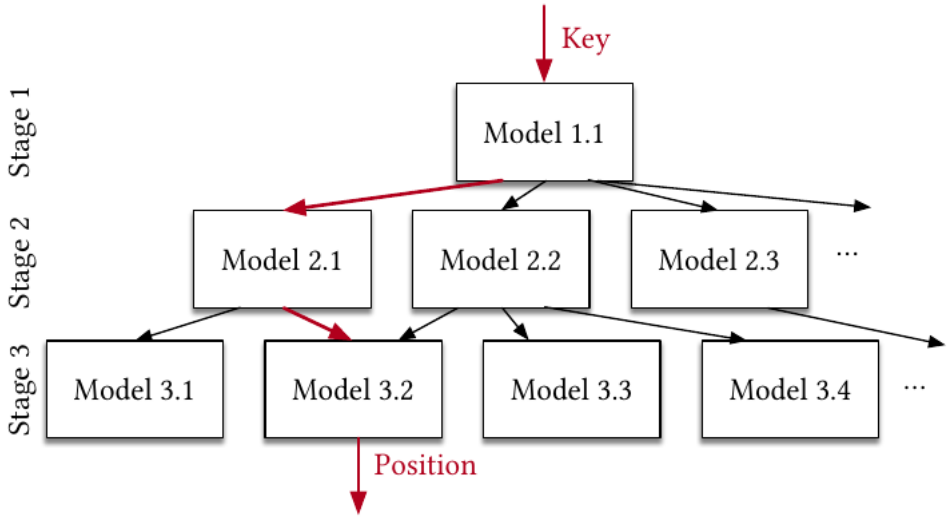

Рисунок 2.3 – Рекурсивная модель индекса

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

Для индексирования строк с помощью ML используют токенизацию — представление строки в виде чисел (NLP).

2.1.2 B^+-деревья

2.1.3 «ИДругие»-деревья

2.2 Индексы на основе хеш-таблиц

2.2.0.1 Из статьи

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

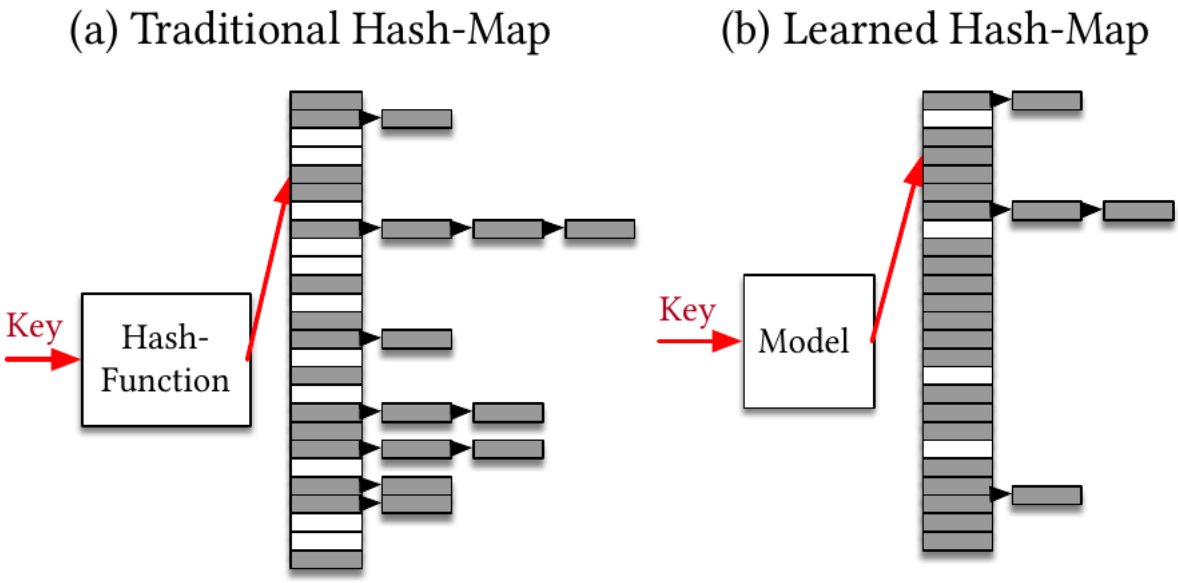

Рисунок 2.4 – Хеш-индексы

Функция 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

2.3.0.1 Из статьи

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 1. Для проверки принадлежности ключа множеству, ключ также подается на вход k хеш-функций. Если какой-либо бит, соответствующий одной из возращенных позиций, равен нулю, то ключ не входит во множество.Из этого следует, что данный алгоритм гарантирует отсутсвие ложноотрицательных результатов.

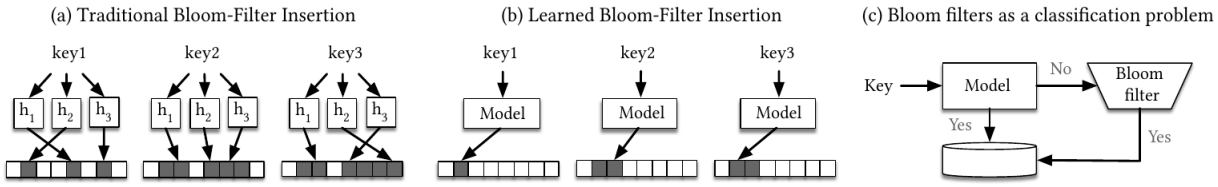

Рисунок 2.5 – Bitmap-индексы

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 возращенных 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. *Григорьев Ю. А.*, *Плутенко А. Д.*, *Плужникова О. Ю.* Реляционные базы данных и системы NoSQL: учебное пособие. — Благовещенск : Амурский гос. ун-т, 2018. — С. 424.
2. *Silberschatz A.*, *Korth H. F.*, *Sudarshan S.* Database System Concepts. — New York : McGraw-Hill, 2020. — С. 1344.
3. *Эдвард Сьоре*. Проектирование и реализация систем управления базами данных. — М. : ДМК Пресс, 2021. — С. 466.
4. *Осипов Д. Л.* Технологии проектирования баз данных. — М. : ДМК Пресс, 2019. — С. 498.