9 Механизм индексации в реляционной СУБД

индекс - вспомогательная структура данных, предназначенная для ускорения некоторых классов запросов. индексирование создаёт отдельную структуру данных, сопоставляющую значения в одном или нескольких столбцах таблицы с соответствующими местоположениями на физическом накопителе, что позволяет базе данных быстро находить строки по конкретному запросу без необходимости сканирования всей таблицы.

индекс - коллекция, состоящая из индексных записей, включающих 2 атрибута:

- индексный ключ - поисковый образ (значения атрибутов, на которых построен индекс)
- информация об объектах - указатели на объекты, содержащие значения атрибутов из индексного ключа

кластерный индекс - используется для сортировки и хранения строк данных на основе первичных ключей.

- существует только один кластерный индекс для каждого отношения.
- кластерный индекс определяет порядок хранения строк отношения.

некластерный индекс имеет структуру, отдельную от строк данных. там содержатся значения ключа и указатель на строку данных. некластерных индексов у отношения может быть много.

структура индексов: хэши, B-деревья (сбалансированное сильно ветвистое дерево)

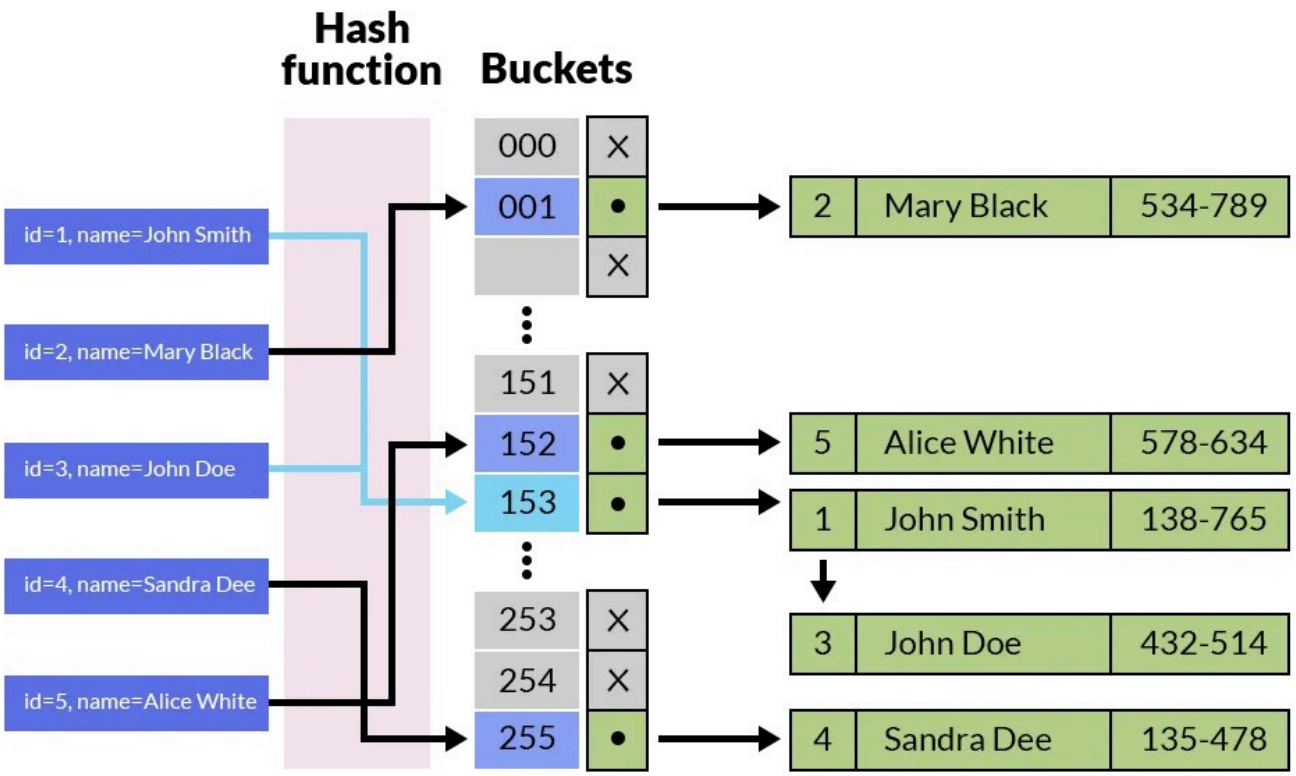
хэш-индекс

Хэш-индекс — это разновидность методики индексирования баз данных, использующая хэш-функцию для сопоставления ключей индекса с местоположениями соответствующих записей данных. Это быстрый метод индексирования для запросов точного соответствия в одном столбце.

Сопоставление ключей индекса с местоположениями соответствующих записей данных позволяет выполнять быстрый поиск и вставки за постоянное время O(1). Однако этот метод плохо работает с запросами диапазонов или частичными совпадениями и может страдать от коллизий, с которыми можно справляться при помощи различных техник разрешения коллизий.

1. разработка хэш-функции. хэш-функция должна быть спроектирована таким образом, чтобы генерировать равномерно распределённое множество хэш-кодов для равномерного распределения записей по корзинам в файле индекса.
2. создание файла хэш-индекса, содержащий набор корзин, каждая из которых соответствует уникальному хэш-коду. корзины содержат указатели на файлы в базе данных, содержащие записи для этого хэш-кода.
3. при выполнении запроса к значению применяется хэш-функция, полученный хэш-код используется для нахождения корзины в файле хэш-индекса. записи с одинаковым хэш-кодом хранятся в одной корзине, поэтому мы можем просто просканировать записи в этой корзине и найти совпадающую запись/записи.

4. чтобы вставить новую запись в хэш-индекс, мы применяем к значению ключа записи хэш-функцию, чтобы сгенерировать его хэш-код, а затем вставляем запись в соответствующую корзину в файле хэш-индекса. если коллизии отсутствуют, вставку можно выполнить за постоянное время O(1), так как нам нужно всего лишь вычислить хэш-код и вставить запись в корзину. если коллизии есть, нам может потребоваться проделать дополнительные операции, например, вставку записи в связанный список в корзине или проверку других корзин, пока не будет найден свободный слот.



недостатки:

- ограниченные возможности поиска (только равенство, т. е. найти все записи, в которых столбец А равен Б. для сортировки походит плохо)
- коллизии (хэш-функция может выдавать одинаковые значения для разных аргументов)
- непредсказуемые требования к размеру хранилища (размер индекса зависит от количества уникальных значений в столбце, а потому его невозможно предугадать)

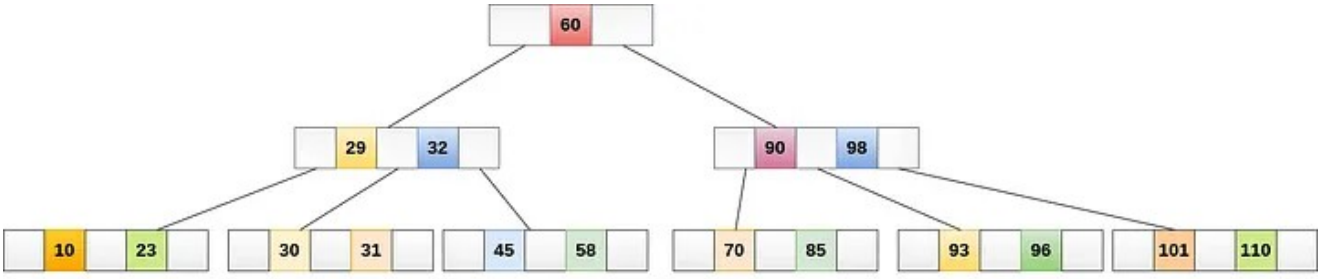
B-дерево

B-дерево — это структура данных самобалансирующегося дерева, которая часто используется в качестве алгоритма индексирования в базах данных. Каждый узел дерева состоит из набора ключей и указателей на дочерние узлы; хранение данных осуществляется в иерархической структуре. Деревья B-узлов упорядочены таким образом, что позволяют быстро выполнять поиск, вставку и удаление данных.

Самое большое преимущество алгоритма B-дерева заключается в минимизации количества дисковых операций ввода-вывода, необходимых для доступа к данным, потому что в B-дереве все узлы-листья находятся на одном уровне, а каждый узел может хранить множество ключей и указателей. Количество ключей и указателей, которое может храниться в узле, определяется параметром, называемым «порядок» дерева.

Алгоритм B-дерева работает следующим образом:

1. **Инициализация**: при создании B-дерева создаётся пустой корневой узел. Параметр, задающий максимальное количество ключей («порядок»), которые могут храниться в каждом узле, управляет B-порядком дерева.
2. **Вставка**: при добавлении нового узла в B-дерево алгоритм сначала подыскивает подходящий узел-лист, в который нужно вставить ключ. B-дерево разделяет заполненный узел-лист на два новых узла и перемещает медианный ключ в родительский узел. Пока не достигнут корневой узел, процесс разделения может распространяться по дереву. Благодаря этой процедуре дерево остаётся сбалансированным, а узлы-листья находятся на одинаковой высоте.
3. **Удаление**: когда ключ удаляется из B-дерева, алгоритм ищет узел, который изначально хранил ключ. Если узел-лист хранил ключ, то ключ извлекается и узел может нуждаться в перебалансировке. Алгоритм удаляет предшествующий или последующий лист после листа-узла, удалив ключ с ним, если ключ обнаружен не в узле-листе.
4. **Поиск**: в процессе поиска ключа в B-дереве алгоритм начинает с корневого узла и рекурсивно движется к веткам, пока не найдёт нужный узел-лист. Метод поиска сравнивает искомый ключ с ключами, содержащимися в каждом узле, а затем использует соответствующий указатель для перехода к дочернему узлу, в котором может находиться ключ. Этот процесс продолжается, пока не будет найден искомый ключ или пока не будет определено, что он отсутствует в дереве.



недостатки:

- излишняя трата ресурсов (каждый узел в B-дереве содержит указатель на родительский и дочерний узлы, поэтому дерево занимает излишне много места)
- сложность (алгоритмы, используемые для вставки, удаления и поиска данных в B-дереве, сложнее по сравнению с другими структурами данных)
- медленные обновления (обновление данных в B-дереве требует множества операций доступа к диску, и этот процесс может быть медленным для больших B-деревьев)

поиск на упорядоченном массиве - $log_2 N$, где N - число элементов.
без индекса - полный перебор (N).

индекс называется плотным, если он содержит ссылки на отдельные объекты индексируемой коллекции.

разреженный индекс хранит ссылку на область памяти. например, одно значение ключа для каждой страницы данных. меньше места, но дольше поиск.

для оптимальной производительности индексы обычно создаются на тех столбцах таблицы, которые часто используются в запросах.
увеличение числа индексов замедляет операции добавления, обновления, удаления строк, поскольку нужно обновлять индексы, а также они занимают дополнительную память.