Структуры внешней памяти

СУБД хранит во внешней памяти следующие разновидности объектов:

- **тела отношений** – основная содержательная пользовательская часть базы данных;
- **служебные отношения-каталоги –** информация, связанная с именованием объектов базы данных и их конкретными свойствами (схема отношения, ограничения).
- **индексы** - управляющие структуры для обеспечения ускоренного доступа (создаются по инициативе пользователя (администратора) или самой СУБД из соображений повышения эффективности выполнения запросов);
- **журнальная информация -** поддерживающая избыточность данных для удовлетворения потребности в надежном хранении данных;
- **дополнительная служебная информация** - информация необходимая СУБД для нормального функционирования (распределения свободной памяти и т.п.).

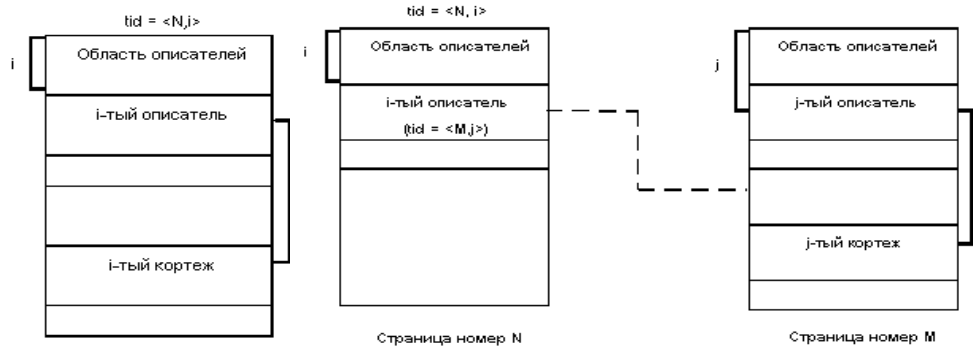
Подходы к физическому хранению отношений

Покортежное хранение отношений.
Единицей физического хранения информации является кортеж.
+ быстрый доступ к целому кортежу,
- много дублирующейся информации,
- лишние обмены с внешней памятью, если нужна часть кортежа.

Хранение отношения по столбцам,
Единицей хранения является столбец отношения с исключенными дубликатами.
+ минимизированы затраты внешней памяти, (за счет исключения дубликатов).
+ возможность использования значений столбца для оптимизации операций соединения.
- сборка целого кортежа (или его части) требует существенных затрат.

Типовой, унаследованной от System R (исследовательская СУБД середины 70-х г. в IBM первый SQL -> IBM DB2), структурой хранения данных является представление внешней памяти как набора сегментов, каждый сегмент состоит из страниц.
Страницы обычно классифицируют на страницы данных и страницы индексной информации. Иногда непосредственно в страницах внешней памяти сохраняют журнальную информацию и «длинные» данные.

Покортежное хранение отношений.
Типовая структура страницы данных:



Каждый кортеж обладает уникальным идентификатором (tid) (tuple identifier), не изменяемым все время существования кортежа. *tuple – кортеж (набор взаимосвязанных величин)*
tid - пара <номер страницы, индекс описателя кортежа в странице>.

Переразмещение кортежа в другую страницу памяти не меняет исходный tid этого кортежа. Меняется только описатель в оригинальной странице, который будет содержать новый tid, указывающий на реальное положение кортежа в новой странице.
Дальнейшее переразмещение кортежа не увеличивают уровень косвенной адресации.

Проблема «длинных» данных
- хранение длинных данных в отдельных (вне базы данных) файлах с заменой данного в кортеже на имя соответствующего файла.
- хранение длинных данных в отдельном наборе страниц внешней памяти, связанном физическими ссылками.
- хранение длинных данных в виде B-деревьев последовательностей байтов в отдельном наборе страниц внешней памяти.

Накладные расходы

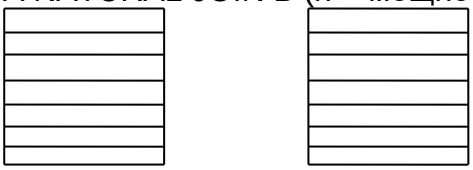
Индексы

Индексы – называют специальные конструкции в СУБД, обеспечивающие механизмы эффективного прямого доступа к произвольному кортежу отношения по значению некоторого ключа и/или последовательного просмотра кортежей в порядке возрастания или убывания значения ключа.

Общей идеей организации индексов является хранение упорядоченного списка значений ключа с привязкой к каждому значению ключа списка идентификаторов кортежей.

При выполнении многих операций языкового уровня требуется сортировка отношений в соответствии со значениями некоторых атрибутов (например NATURAL JOIN).

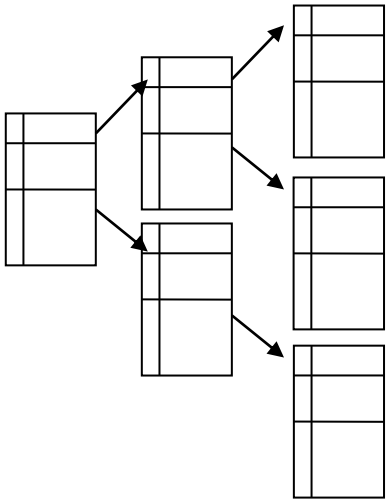
A NATURAL JOIN B (n = мощность отношения A и B)



При отсутствии индекса сложность операции n^2
При наличии индекса в отношении B сложность операции n*log(n)
При наличии индекса в отношении A и B сложность операции 2*n

Различные способы организации индекса отличаются **способом поиска ключа**.

Организация индексов на основе техники B-деревьев



B-дерево - это сбалансированное сильно ветвистое дерево во внешней памяти.

Сбалансированность означает, что длина пути от корня дерева к любому его листу практически одинакова.
Ветвистость дерева - это свойство каждого узла дерева ссылаться но большое число узлов-потомков.

Физическая организация B-дерева представляет собой древовидная мульти-списочная структура внутренних и листовых страниц внешней памяти.

Листовая страница имеет следующую структуру:
(< key_i , $TIDs_i$ > …)

- $key_1 < key_2 < \ldots < key_n$
- $TIDs_i = \{ tid_{i1}, tid_{ij}, tid_{im} \}$ – tid всех кортежей в которых key_i
- все листовые страницы связаны одно- или двунаправленным списком.

Внутренняя страница имеет следующую структуру:
(< key_i , N_i >…)

- $key_1 <= key_2 <= \ldots <= key_n$
- N_i – ссылка на внутреннюю или листовую страницу в которой находятся ключи с значениями лежащими в интервале [key_i … key_{i+1}]

Поиск в B-дереве - это прохождение от корня к листу в соответствии с заданным значением ключа. За счет сильной ветвистости и сбалансированности дерева, для выполнения поиска по любому значению ключа потребуется проход на одинаковую небольшую глубину – т.е. фактически одно и то же (небольшое) число обменов с внешней памятью.
В сбалансированном дереве, с размером внутренней страницы в **n** ключей, для хранении **m** записей потребуется глубина $\mathbf{log_n m}$.

Автоматическое поддержание свойства сбалансированности B+ деревьев

При занесении новой записи выполняется:

- Поиск листовой страницы.
- Считывание листовой страница в буфер (размер больше размера страницы) и его модификация (вставка нового значения).
- Если после вставки размер используемой части буфера не превосходит размера страницы, то операция завершается и буфер может быть немедленно вытолкнут во внешнюю память.
- Если возникло переполнение буфера, то выполняется операция расщепление страницы. Запрашивается новая страница внешней памяти и используемая часть буфера разбивается пополам и вторая половина записывается во вновь выделенную страницу.
- Для обеспечения доступа к заново заведенной страницы, модифицируется внутренняя страница, являющуюся предком ранее найденной страницы. В соответствующее место осуществляется вставка ссылки на вновь созданную страницу. Операция осуществляется в буфере оперативной памяти где снова может произойти переполнение и данная стадия может повторится на более высоком уровне дерева.
- Предельным случаем является переполнение корневой страницы B-дерева. В этом случае она тоже расщепляется на две, и заводится новая корневая страница дерева, т.е. глубина дерева увеличивается на единицу.

Автоматическое поддержание свойства сбалансированности B+ деревьев
При удалении записи выполняются следующие действия:

- Поиск записи по ключу.
- Считывание листовой страница в буфер и реальное удаление записи.
- Если после выполнения удаления размер занятой в буфере области оказывается таковым, что его сумма с размером занятой области в листовых страницах, являющихся предыдущей или последующей для данной страницы, больше, чем размер страницы, операция завершается.
- Иначе производится слияние с предыдущей или последующей страницей, т.е. в буфере производится модификация образа соседней страницы, содержащей общую информацию из данных страниц. Ставшая ненужной листовая страница заносится в список свободных страниц, а также модифицируются сквозные ссылки листовых страниц.
- Для обеспечения прекращения доступа к удаленной листовой странице в оперативной памяти модифицируется образ внутренней страницы, являющейся предком ранее существовавшей листовой страницы. После удаления ссылки в оперативной памяти снова может возникнуть ситуация когда будет возможно слияние модифицированной страницы с соседней и повторение данной стадии на более высоком уровне иерархии.
- Предельным случаем является полное опустошение корневой страницы дерева, которое возможно после слияния последних двух последних потомков корня. В этом случае корневая страница освобождается, а глубина дерева уменьшается на единицу.

Для уменьшения проблем потери эффективности при слишком частом выполнении операций расщепления и слияния, применяются более сложные приемы, в частности:

- **упреждающие расщепления**,
  т.е. расщепления страницы не при ее переполнении, а несколько раньше, когда степень заполненности страницы достигает некоторого уровня;
- **переливания**,
  т.е. поддержание равновесного заполнения соседних страниц;
- **слияния 3-в-2**,
  т.е. порождение двух листовых страниц на основе содержимого трех соседних.

Хеширование

Общей идеей методов хеширования является применение к значению ключа функции свертки (хэш-функции), вырабатывающей значение меньшего размера. Свертка значения ключа затем используется для доступа к записи.
i = f(X), X – что угодно, i – обычно int индекс в некотором фиксированном интервале

В самом простом, классическом случае, свертка ключа используется как индекс элемента в массиве, содержащем множество пар ключей и tid-ов. Больше одной пары в элементе массива (которые иногда называют «**цепочки переполнения**») появляется в случае возникновения коллизий – когда различным значениям ключа соответствует одно и то-же значение свертки.

Основным требованием к хэш-функции является равномерное распределение значение свертки.

Если таблица заполнена слишком сильно или переполнена, но возникнет слишком много цепочек переполнения, и главное преимущество хеширования - доступ к записи почти всегда за одно обращение к таблице - будет утрачено.

Главным ограничением этого метода является фиксированный размер таблицы. Расширение таблицы требует ее полной переделки на основе новой хэш-функции (со значением свертки большего размера).

В СУБД в этом случае часто применяют техники совмещающие использование B+ деревьев и хеширования.

Журнальная информация

Журнал обычно представляет собой чисто последовательный файл с записями переменного размера, которые можно просматривать в прямом или обратном порядке. Обмены производятся стандартными порциями (страницами) с использованием буфера оперативной памяти.
В грамотно организованных системах структура (и тем более, смысл) журнальных записей известна только компонентам СУБД, ответственным за журнализацию и восстановление. К ведению файла журнала предъявляются особые требования по части надежности. В частности, обычно стремятся поддерживать две идентичные копии журнала на разных устройствах внешней памяти.

Служебная информация

- внутренние каталоги, описывающие физические свойства отношений: число атрибутов, их размер, типы данных, описание индексов…
- описатели свободной и занятой памяти в страницах отношения
- информация о связывании страниц внешней памяти для одного отношения.